VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Hardwarové řešení inteligentního domu

Hardware Solution for Intelligent Household

2011 Bc. Patrik Dubec

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2011 .....

Rád bych na tomto místě poděkovala všem, kteří mi s prací pomohli. Především svému vedoucímu Ing. Michalu Krumniklovi za odbornou pomoc a trpělivost při vypracovávání diplomového projektu. Dále našemu klientovi, který s námi více než ochotně spolupracoval a umožnil nám otestovat výsledek práce u něj doma.

Abstrakt

Tato práce se zabývá návrhem a implementací hardwarového základu pro inteligentní domácnost se záměrem usnadnit život tělesně postiženým. Nedílnou součástí je tvorba softwarového API, které umožní integrovat tento základ do jakéhokoli systému. API musí umožnit přístup k ovládání jednotlivých periferií pomocí bezdrátových technologií. Použití API demonstruji v mobilní aplikaci, která slouží jako dálkové ovládání pro systém. Při návrhu se opíráme o metodologii Unified Process (UP). Pro celkové řešení je použito technologií firmy Microsoft, jako programovacího jazyka C#, aplikačních rámců .NET Framework a .NET CF, operačního systémů Windows Embedded. Závěrečnou etapu tvoří nasazení a otestování aplikace přímo u klienta.

Klíčová slova: inteligentní domácnost, Windows phone 7, Windows standard embedded 7, webová služba, Windows služba, Windows Comunication Foundation, Silverlight, .NET Framework

Abstract

This project is focused on design and developing of a hardware layer that will be the backbone for intelligent household. The main purpose is an effort to ease conditions of living for people with some kind of handicap. Integral part of project is developing of standard API layer that allows integrating the hardware solution to any kind of a software information system (IS). The API has to provide an option of handling with components via a wireless approach. Using of the API is demonstrated in mobile application that serves like remote controller. There is used methodology of Unified Process (UP) by designing of the solution. There are used Microsoft technologies like programing language c# frameworks .NET Framework, .NET CF, operating system Windows Embedded for overall purposes. The final part of project is deploying and testing of our solution on the client's side.

Keywords: intelligent household, Windows phone 7, Windows standard embedded 7, web service, Windows service, Windows Comunication Foundation, Silverlight, .NET Framework

Seznam použitých zkratek a symbolů

stand-alone	–	Stand-alone aplikace, je aplikace, která pro svůj běh nepotřebuje práci v síti
OS	–	OS je zkratka pro operační systém
WP7	–	je zkratka pro mobilní operační systém Windows Phone 7 firmy Microsoft
WES7	–	je zkratka pro operační systém Windows Standard Embedded 7 firmy Microsoft
TDP	–	Thermal Design Power - lze přeložit jako navržený tepelný výkon. Tato hodnota představuje nejvyšší možný tepelný výkon, který musí chlazení CPU umět odvést
WCF	–	je zkratka pro technologii Windows Comunication Foundation sloužící k tvorbě distribuovaných aplikací
.NET CF	–	je zkratka pro aplikační rámec .NET Compact Framework pro použití v mobilních zařízeních
UP	–	je zkratka pro Unified Process, což je metodika (rámec) popisující unifikovaný proces vývoje aplikací

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Cílem této práce je navrhnout a vytvořit hardwarový základ systému sloužícího pro ovládání inteligentní domácnosti. Důležitou součástí je tvorba standardního API umožňující na tento základ navázat, jak tomu bude ve své diplomové práci činit Bc. Jan Plucar, který ve svém IS zaintegruje veřejné rozhraní sloužící pro ovládání domácnosti. Hlavním přínosem by mělo být usnadnění a zlepšení kvality života tělesně postiženým lidem, pro které je tento projekt primárně určen. Řešení by mělo nabízet co možná největší modularitu a robustní architekturu pro budoucí možnost rozšíření funkcionality s minimálním vlivem na stávající systém. Naše zařízení bude schopno bezdrátové komunikace s dálkovým ovládáním.

Po celou dobu realizace projektu se budeme řídit metodikou UP popsanou v knize [1] (Unified Software Development Process - unifikovaný proces vývoje softwaru), která nám poskytne rámec pro uchopení celé problematiky vhodným způsobem. UP nám pomůže definovat aspekty systému říkající *kdo* (workers), *kdy*(workflows) a *jak*(artefacts) bude se systémem moci pracovat.

Tato metodika je založena na návrhu a postupném (iterativním) vývoji robustní architektury našeho systému. Iterativní přístup pro nás znamená dekompozici celkového problému na dílčí (snáze řešitelné) problémy, které se budou postupně připojovat k již stávajícímu řešení (jedná se o tzv. inkrementy).

Nejprve si tedy v kapitole Analýza a návrh analyzujeme zadání, tím získáme funkční a nefunkční požadavky a dostatečně podrobnou představu o tom, co vlastně budeme od hotového systému očekávat. Dále položíme základ architektury. V této fázi si ujasníme rozdělení celého systému na dílčí komponenty. Budeme také muset učinit zásadní rozhodnutí, co se výběru vhodné platformy týče. V poslední fázi si pro zvolenou platformu vytvoříme funkční obraz operačního systému Windows Embedded.

V kapitole implementace se podíváme na to, jak se náš návrh překlopí ve fungující systém. Podíváme se na některé ze zajímavých implementační postupů, které byly při vývoji použity. Také se zde podíváme na některá technická úskalí, na která jsme pří implementaci narazili a povíme si, jakým způsobem bylo třeba je řešit.

V kapitole nasazení shrneme, jak úspěšně dopadl pilotní provoz u klienta. Dozvíte se zde také, k jakým úpravám jsme přistoupili na základě zpětné vazby, kterou jsme během provozu získali.

2 Analýza a návrh

V této kapitole se budeme soustředit převážně na sběr a analýzu požadavků. K tomu využijeme osvědčené metody a postupy známé z disciplíny softwarového inženýrství. Na základě získaných informací zahájíme tvorbu analytického modelu, který zachytí požadované chování programového díla. Tento model se zaměřuje na to, *co* systém musí udělat, nezabývá se detaily týkajícími se způsobu, *jakým* to udělá. Tuto otázku přenechává aktivitě návrh (design). V této práci pro stručnost tyto dvě samostatné etapy ne zcela správně sloučíme do jedné kapitoly.

2.1 Zadání klienta

> Chtěla bych si nechat vytvořit systém, který by mi umožnil řídit a ovládat některá zařízení v domě, kde bydlím. Díky mé částečné fyzické indispozici mám problémy s některými běžnými úkony. Proto bych si přála, aby mi výsledný produkt usnadnil alespoň některé z těchto činností. Problémy mi činí například dostatečně rychle reagovat na zvonění lidí u dveří. Často se mi stává, že než stihnu dojít ke dveřím, tak návštěva odejde. Obecně mi činí zvýšenou námahu, když musím vstávat. Ocenila bych tak například možnost ovládání světel v domě, zatáhnutí žaluzií u oken. Systém by mohl automaticky ovládat topení podle teploty v místnosti a udržovat tak nastavenou teplotu. Občas se mi stane, že potřebuji pomoc druhé osoby, systém by tedy měl umět oznámit někomu v domě, že potřebuji pomoct.
> K ovládání bych chtěla používat dálkový ovladač, který bych mohla mít neustále u sebe. Ovladač se mi musí pohodlně ovládat (veliká tlačítka). Pro zbytek rodiny by mělo být přístupné nějaké centrální ovládání. Zároveň celé řešení nesmí být příliš drahé.

2.2 Specifikace požadavků

Důležité je hned z počátku zjistit, **co** bude od systému očekáváno a co již nikoli, tzn. striktně definovat hranice systému. Při tvorbě specifikace požadavků budeme aktivně konzultovat potřeby a navrhovaná řešení přímo s budoucím uživatelem systému, který se svolil také k nasazení a otestování hotového řešení. To nám významně pomůže se co nejlépe vcítit do požadavků klienta (popř. dalších potencionálních klientů), a tak zkvalitnit výsledný produkt.

Prvním krokem při tvorbě nového systému je vždy získání požadavků (zadání) od klienta. Ani zde tomu nebude jinak. V počáteční fázi tento krok obnášel schůzku se zákazníkem, kde jsme od něj získali zevrubné zadání ve formě krátkého článku, který jste si již mohli přečíst na začátku této kapitoly. Pro upřesnění ještě doplním způsob postižení klienta. Klient má omezenou pohyblivost z důvodu chybějící části jedné dolní končetiny. Dále má v lokti amputovánu jednu horní končetinu a na druhé horní končetině má pouze dva prsty.

Pro přehlednost zde uvedu hlavní body tohoto textu. Díky snížené pohyblivosti by klientovi podle jeho slov pomohlo, kdyby si mohl:

- ráno rozsvítit světlo v pokoji, aniž by musel vstát po tmě z postele
- nastavit minimální teplotu v místnosti. Pokud by došlo k poklesu teploty pod stanovenou mez, systém by měl spustit ústřední topení.
- v případě potřeby přivolat člena rodiny (ošetřovatele) z jiné místnosti domu.
- často klientovi trvá, než dojde otevřít dveře, pokud někdo zvoní u dveří, proto by uvítal možnost zjistit, kdo zvoní a případně návštěvníka dálkově vpustit do domu.
- uvítal by přenosné dálkové ovládání s velikým displayem a snadným intuitivním ovládáním.
- ovládat celý systém pomocí dotykové obrazovky umístěné poblíž zařízení (bude sloužit pro ostatní členy domácnosti).
- cena realizace nesmí být příliš vysoká.

Na první pohled se může jevit, že požadavků není mnoho. Je to z části zapříčiněno tím, že zákazník má vždy z počátku jen mlhavé představy o tom, co vlastně od systému bude v konečné fázi chtít. Je proto na analytikovi, aby se zákazníkem vykomunikoval veškeré detaily na základě jeho prvotních požadavků. Při bližším prozkoumání tedy zjistíme, že značná rozmanitost požadavků, získaných přímo ze zadání, nás nutí k tvorbě velice komplexního systému, ve kterém bude muset spolupracovat množství samostatných modulů.

Pomocí metody **analýza podstatných jmen a sloves** odhalíme v zadání hlavní aktéry, analytické třídy a jejich atributy a dále funkční a nefunkční požadavky. Jak jsem již uvedl, postupujeme podle metodiky UP, která spočívá v inkrementálním postupu. To znamená, že při první analýze neodhalíme ještě veškeré požadavky. Funkční požadavky výborně poslouží jako podklad pro tvorbu případů užití, které jsou pro následnou implementaci stěžejní. Vzhledem k omezenému rozsahu zde nebudu uvádět výsledky všech iterací, ale pouze jejich finální podobu.

2.2.1 Aktéři systému

Aktér specifikuje roli, kterou určitá entita přijímá v okamžiku, kdy začíná daný systém bezprostředně používat. Může vyjadřovat roli uživatele, roli dalšího systému, který se dotýká hranic našeho systému. Již zde jsme schopni odhalit dva hlavní aktéry systému:

Uživatel mobilního zařízení bude systém ovládat pomocí dálkového ovladače. Nebude mít přístup ke všem funkcím systému. To především z důvodu malé velikosti displaye, kde by se některé konfigurace prováděly obtížně. Dále je zde snaha o co nejintuitivnější ovládání. Přílišné množství málo využívaných oken zbytečně ztíží přístup k funkcím zásadním.

Uživatel dotykového LCD bude systém ovládat z připojeného dotykového LCD panelu. Bude mít k dispozici veškeré nabízené možnosti, včetně nastavení.

Čas Pro úplnost zde uvádíme i aktéra *čas*. To proto, protože očekáváme, že systém bude umět automaticky vyvolat některé události v závislosti na určitém časovém bodě.

Softwarové inženýrství není exaktní věda, to znamená že velkou roli při analýze hraje subjektivní vnímání problému analytikem. Někteří by zde ještě uvedli také ovládané periferie, které se systémem budou aktivně komunikovat a poskytovat jim data různého druhu. Mohli bychom je tedy považovat za vedlejší aktéry. My zde však budeme tyto periferie vnímat jako interní součásti systému a tudíž je za aktéry považovat nebudeme.

2.2.2 Funkční požadavky

Funkčním požadavkem je formulace toho, co by měl systém dělat - popisuje požadovanou funkci systému. Jako zdroj těchto požadavků slouží zadání zákazníka. Nebudu zde uvádět veškeré požadavky, ale pouze nejdůležitější z nich. U funkčních požadavků si také zavedeme prioritizaci podle následujícího klíče. Nejdůležitější požadavky používají frázi *Systém* ***bude*** *...*, a dále s klesající prioritou *Systém* ***by měl*** *...*, *Systém* ***by mohl*** *...* a nakonec nejnižší prioritu značíme ***Chceme****, aby systém ...*. Prioritou požadavku tak nenápadně říkáme, které vlastnosti jsou pro systém klíčové a budou se tak implementovat jako první. Požadavky s nižší prioritou je možné implementovat až v další verzi.

1. Systém bude umět rozsvítit světlo.
2. Systém bude umět promítnout obraz z kamery.
3. Systém bude umět zjistit teplotu v místnosti.
4. Systém bude reagovat na příkazy dálkového ovládání.
5. Systém bude reagovat na příkazy z dotykové obrazovky.
6. Systém bude umět autorizovat požadavek dálkového ovládání.
7. Systém bude umět vydat výstražné znamení.
8. Systém by měl umět regulovat teplotu.
9. Systém by mohl umět otevřít dveře.
10. ...

2.2.3 Nefunkční požadavky

Vedle funkcionality systému nás také zajímají omezení kladená na systém, kterým se říká nefunkční požadavky. Nefunkční požadavek je tedy omezující podmínka říkající, jaké postupy a technologie mají být při řešení použity. Při analýze zadání zjistíme, že takových restrikcí máme hned několik.

1. Embedded server poběží na operačním systému Windows Embedded.
2. GUI dálkového ovládání bude realizováno podle specifických požadavků klienta.
3. Celkové řešení musí být finančně šetrné.
4. Embedded server bude nabízet veřejné rozhraní pro ovládání přes síť, které bude využito ve více aplikacích.
5. ...

2.3 Případy užití (Use Cases)

Na základě této prvotní analýzy, díky které jsme odhalili funkční požadavky jsme vypracovali možné scénáře případů užití, které jsme opět představili klientovi. Na základě zpětné vazby jsme jim vdechli konečnou podobu. Jednotlivé případy užití si můžete prohlédnout na obrázku 1 v diagramu případu užití. Pro stručnost zde nebudu uvádět veškeré detailní specifikace všech těchto use casů.

2.3.1 Případ užití - ovládání osvětlení

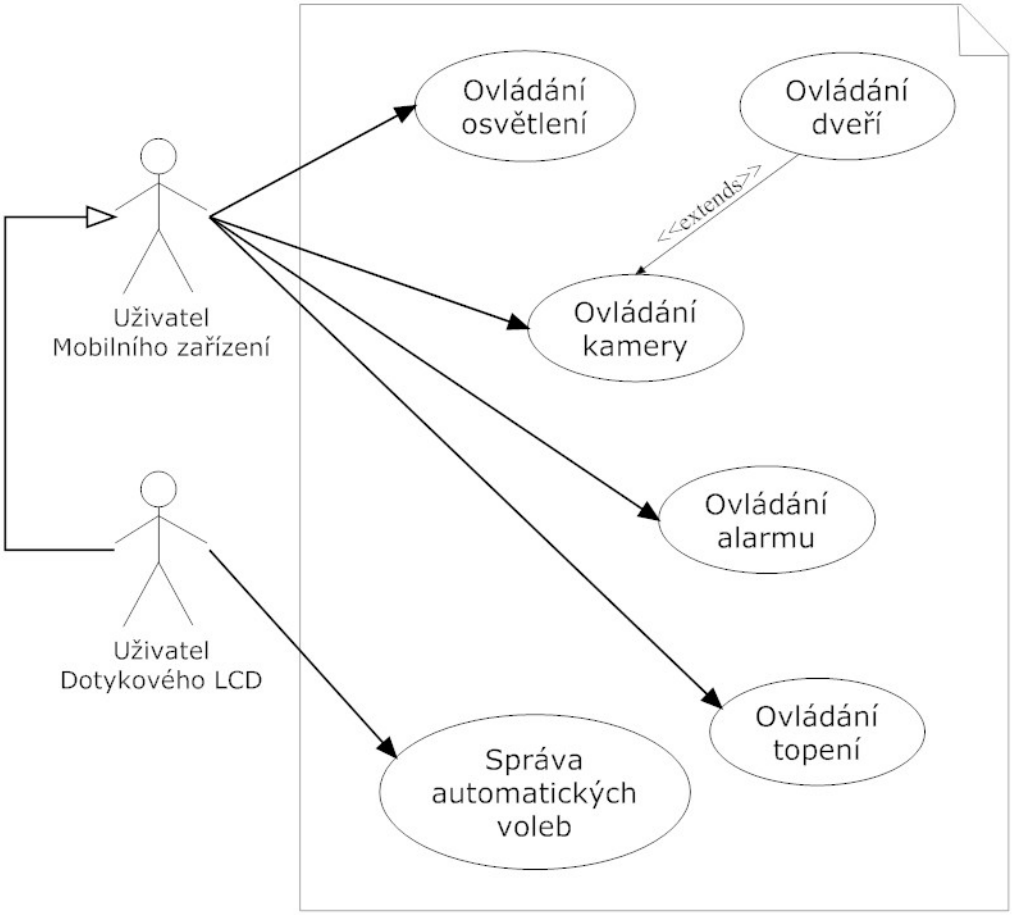
Tento případ užití vznikl na základě scénáře (user story), kde klient vylíčil svou představu o ovládání osvětlení. Výsledný use case si můžete prohlédnout v následující tabulce 1.

2.4 Analytické třídy

Jak jsem již dříve předeslal, pomocí metody *analýza podstatných jmen* ze zadání získáme kandidáty na analytické třídy a atributy. V druhé fázi z těchto kandidátů vybereme třídy. Entity, které nesplňují požadavky třídy, se stanou atributy. Analytické třídy jsou takové entity, jejichž instance reprezentují objekty z problémové domény (mapují pojmy skutečného světa). To znamená, že se zde nebudeme nebudeme pokoušet modelovat např. třídy pro práci s databází. To je již úkolem vývojové etapy návrh. V následujícím obrázku 2 se můžeme podívat na některé analytické třídy, které se v našem projektu objevují.

2.5 Realizace případů užití

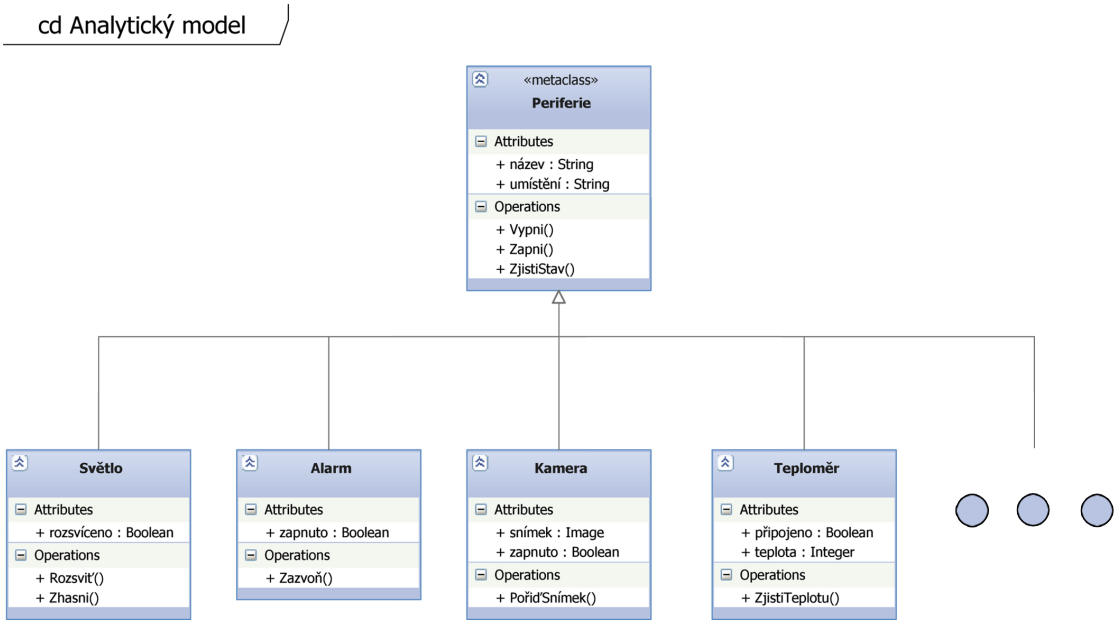
V současné době již tedy máme hotovy analytické třídy, které spolu se specifikací případů užití tvoří podklad pro tvorbu sekvenčních diagramů. Sekvenční diagramy znázorňují interakce mezi čárami života jako časově uspořádanou posloupnost událostí. Pomohou nám tedy zmapovat, jak budou jednotlivé objekty mezi sebou komunikovat pro dosažení uživatelských cílů. Mohli bychom rovněž použít aktivitní diagramy, které by nám navíc umožnily zachytit zodpovědnosti jednotlivých modulů. Pro dostatečné pochopení si však vystačíme s diagramy sekvenčními.



Obrázek 1: Diagram případů užití

Případ užití - ovládání osvětlení
ID: 1
Stručný popis: Souhrný postup pro rozsvícení/zhasnutí světla
Hlavní aktéři: Uživatel Mobilního zařízení, Uživatel dotykové LCD
Vedlejší aktéři: žádní
Vstupní podmínky: Služba systému pro ovládání domácnosti je spuštěna. Aktér Uživatel mobilního systému je ověřen vůči systému.
Hlavní scénář: 1. Případ užití začíná volbou uživatele pro ovládání světel. 2. Systém na základě konfigurace zobrazí všechna dostupná připojená osvětlovací zařízení. 3. Uživatel zvolí osvětlovací bod, který chce ovládat (a) Pokud je osvětlení ve stavu „ON" systém nabídne volbu pro zhasnutí. (b) Pokud je osvětlení ve stavu „OFF", systém nabídne volbu pro rozsvícení. 4. Uživatel zvolí možnost Zhasni. 5. Systém zhasne příslušné světlo
Výstupní podmínky: Systém zhasne/rozsvítí světlo Systém zaznamená stav zařízení (světla)
Alternativní scénáře: 4. a Uživatel zvolí možnost Rozsvit'. 5. a Systém rozsvítí světlo.

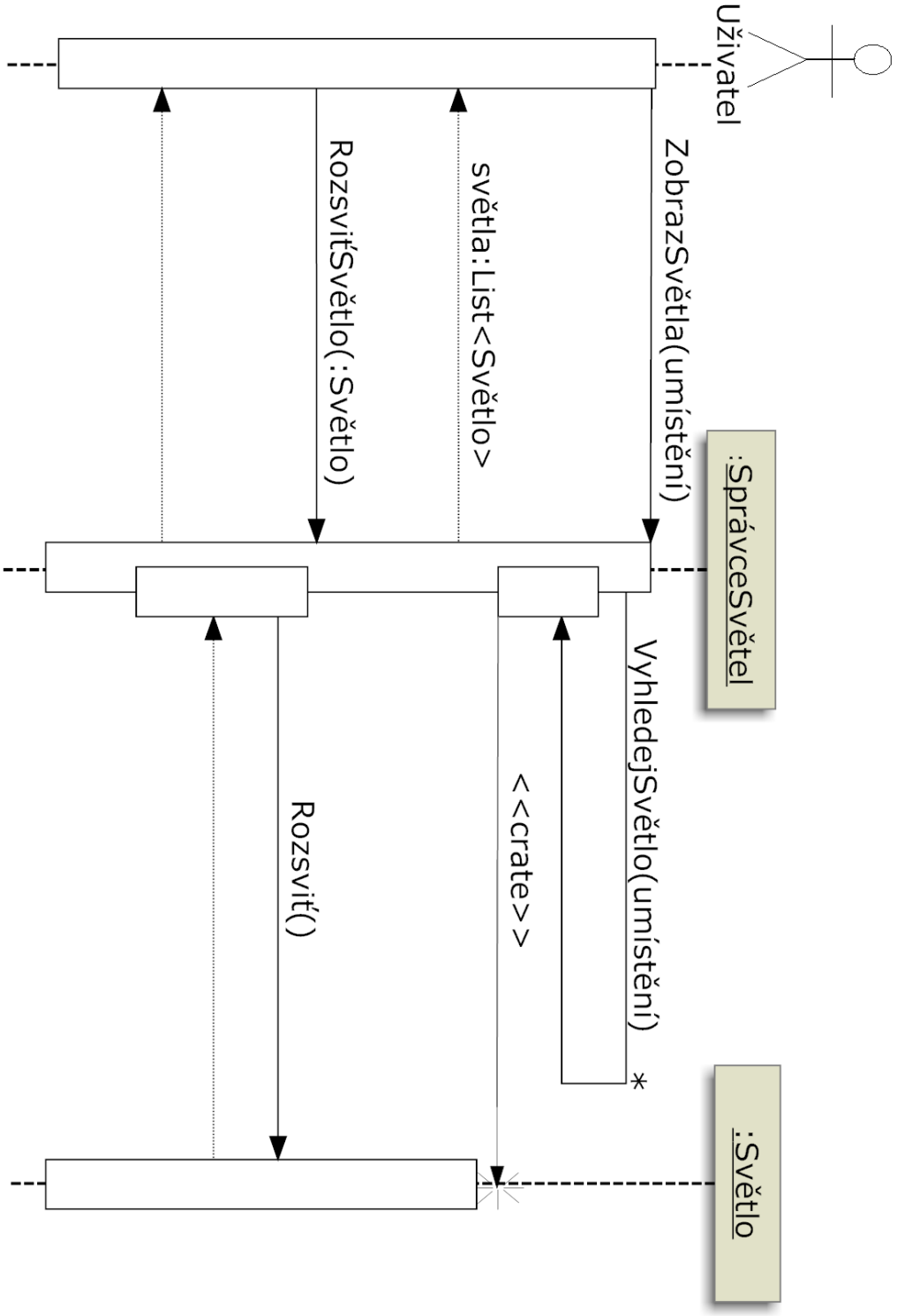
Tabulka 1: Případ užití - ovládání osvětlení



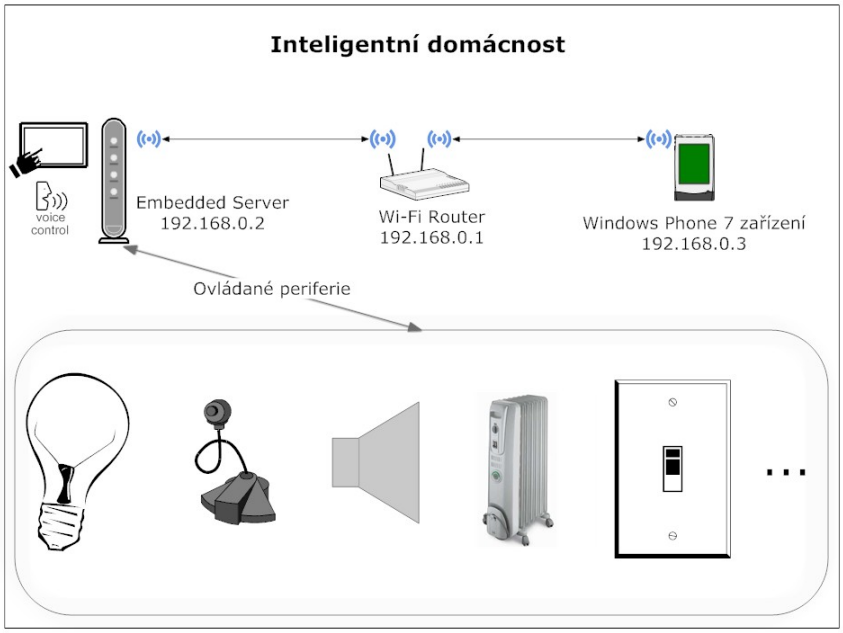
Obrázek 2: Analytické třídy

2.5.1 Sekvenční diagram - ovládání světel

Následující diagram 3 znázorňuje jak probíhá komunikace mezi třídami, pokud chce uživatel rozsvítit světlo ve zvoleném pokoji.



Obrázek 3: Sekvenční diagram - ovládání světel



Obrázek 4: Celková architektura systému

2.6 Návrh

V této chvíli již přesně víme, co má výsledná aplikace dělat. Máme zmapovánu problémovou doménu, víme jaké objekty se v systému vyskytují, jaké funkce bude systém nabízet. Zbývá nám tedy zodpovědět otázku, *jak* to vše systém bude dělat. V následujících několika odstavcích si rozebereme celkovou struktury vyvíjeného systému.

2.6.1 Návrh architektury

Nevytváříme triviální stand-alone aplikaci, ale stojíme před poměrně složitým úkolem, a tak je třeba si hned na začátku jasně definovat stabilní architekturu, ze které bude naše řešení vycházet. Každá chyba v tomto návrhu se v pozdější části může stát velmi drahou (jak časově, tak finančně) na opravu. Jak jsem již naznačil, celé řešení se bude skládat z více „samostatných" aplikací (modulů), které budou mezi sebou muset komunikovat a tím tak uspokojit potřeby uživatele. Jednotlivé moduly poběží na více hardwarových zařízeních a tím pádem budou muset využívat různé technologie, což nás staví před problém, jak co nejefektivněji zajistit komunikaci mezi těmito zařízeními.

Hlavní částí celého systému bude embedded zařízení, které bude přímo komunikovat s ovládanými periferiemi, jak je patrné z obrázku 4. Toto embedded zařízení (nazvěme jej server), bude možné řídit buď pomocí připojené dotykové obrazovky, nebo dálkově pomocí mobilního zařízení (např. chytrého telefonu). Pro komunikaci mezi mobilním telefonem a serverem se jako optimální komunikační kanál jeví domácí Wi-Fi síť. A to především díky síle a pokrytí signálu, což u jiných běžně používaných bezdrátových tech-

nologií, jako např. Bluetooth(2.0), nebo IrDA nenajdeme. Srovnání těchto bezdrátových technologií blíže uvádím později v kapitole 2.6.3.

2.6.1.1 Architektura aplikace embedded serveru Nejprve je třeba si ujasnit, co všechno budeme od tohoto zařízení potřebovat a jakým způsobem docílit kýžené funkcionality.

Server bude muset zastávat následující role:

1. Bude muset obsluhovat (ovládat, nebo zjišťovat jejich stav) připojené periferie
2. Bude muset být schopen komunikovat s uživatelem skrze dotykovou obrazovku
3. Bude muset umět reagovat na požadavky mobilního zařízení sloužícího jako dálkové ovládání

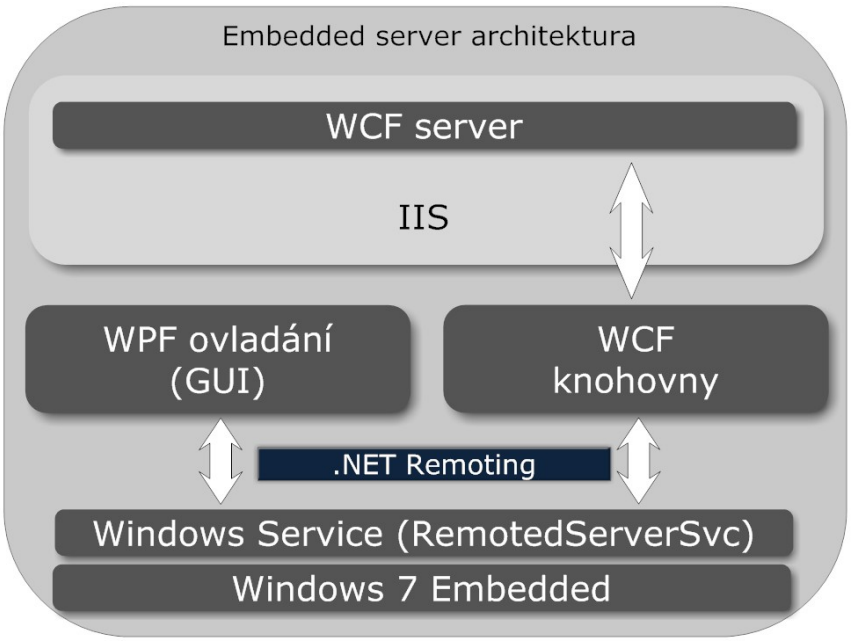
Uvedené požadavky s sebou přináší jistá omezení. Podrobněji se budu touto problematikou zabývat v sekci 2.8, kde objasním, proč jsme k řešení zvolili ty které postupy a technologie.

Jako operační systém tedy bude použit Windows Embedded Standard 7 SP1, který nám poskytne nezbytný základ pro běh ostatních modulů naší aplikace. Pro veškeré dílčí aplikace je využita vícevrstvá architektura, která zajišťuje jak snazší udržovatelnost kódu, tak logické oddělení výkonného kódu (Application logic) od vrstvy prezentační (GUI). Rád bych ještě zdůraznil, proč je obzvlášť v tomto systému důležité dodržovat toto dělení. Znovu připomínám, že se při implementaci setkáme s různými technologiemi napříč celým systémem. Z toho plyne, že stejná funkcionalita může být, a také bude, vyžadována na více místech (aplikace pro dotykovou obrazovku, mobilní zařízení). Dalším faktorem, ke kterému je třeba přihlížet je, že se aplikace vyvíjí pro potřeby tělesně postižených lidí. V reálu tedy jistě narazíme na požadavek vytvořit GUI na míru podle specifických potřeb zákazníka. Není tedy myslitelné, aby se udržovalo více verzí modulů programu, kde by se opakoval kód aplikační logiky.

Jak je patrné z obrázku 5, do běhového prostředí bude nainstalována služba (Windows Service) - **Remoted Server Service**. Jejím posláním bude plnit dvě základní povinnosti. Bude integrovat ovládání a obsluhu veškerých hardwarových periferií. Jak přesně popíši v kapitole . Druhou povinností této stěžejní služby bude nabízet veřejné rozhraní pro komunikaci, a tím samotné ovládání periferií.

2.6.1.2 Architektura aplikace mobilního zařízení Mobilního klienta bude v našem případě reprezentovat chytrý mobilní telefon s operačním systémem Windows Mobile (OS). Původně jsme zamýšleli používat dnes nejvíce rozšířenou verzi tohoto OS, tedy Windows Mobile 6.5. Proč jsme nakonec od této verze přešli na nejnovější verzi mobilního operačního systému Windows Phone 7 (WP7), který je založen na zcela novém a odlišném jádře, popíši v kapitole 2.9.

Mobilní klient, jehož struktura je znázorněna na obrázku 6, bude uživateli sloužit jako dálkové ovládání, pomocí kterému bude možné ovládat veškeré periferie (zařízení). Jen těžko si asi můžeme představit, že telefon bude tato zařízení ovládat přímo. Proto



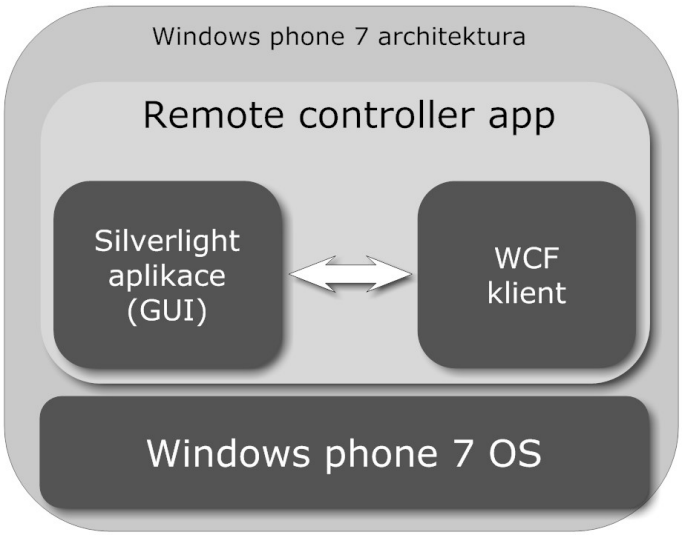
Obrázek 5: Architektura aplikace embedded serveru

bude nutné posílat požadavky skrze naše embedded zařízení (server), od kterého musíme nutně vyžadovat, aby nabízel „veřejné" rozhraní, které nám tento způsob komunikace umožní. Uvádím, že rozhraní serveru musí být veřejné, to prosím berte s rezervou. Musíme samozřejmě implementovat mechanismy pro ověření zdroje požadavku, aby nemohlo (s trochou nadsázky) dojít k situaci, že náhodný kolemjdoucí získá přístup k ovládání.

Aplikace sama o sobě nebude přinášet žádnou novou funkcionalitu pro celkový systém. Telefon má sloužit jako pasivní klient (dálkové ovládání) využívající **push** přístup. To znamená, že server nebude sám od sebe aktivně komunikovat s tímto klientem, ale právě naopak. Mobilní klient na příkaz uživatele sám kontaktuje server se svým požadavkem a poté zpracuje příchozí odpověď. Mohli bychom tedy říct, že jde o tenkého klienta (thin client).

Samotná aplikace bude opět striktně rozdělena do více logických vrstev. Budeme implementovat vrstvu, která se bude starat o komunikaci se serverem, tj. posílání požadavků a převzetí adekvátních odpovědí. Požadavky budou samozřejmě vznikat z interakce uživatele s vrstvou uživatelského rozhraní (GUI), kde budou příchozí odpovědi adekvátním způsobem reprezentovány. Zde opět narážíme na nutnost separovat GUI vrstvu od aplikační logiky. Musíme mít stále na paměti, že aplikace je vyvíjena pro potřeby tělesně postižených lidí. Proto se v budoucnu bude pravděpodobně tvořit i další grafické rozhraní, ve kterém s výhodou použijeme stávající funkcionality.

Nakonec tedy ještě budeme potřebovat vrstvu, která bude zachytávat události (interakce uživatele s GUI) a předávat požadavky vrstvě pro komunikaci se serverem. Na



Obrázek 6: Architektura aplikace mobilního telefonu

druhou stranu bude přebírat odpovědi ze serveru a vhodným způsobem je zpracovávat pro vrstvu uživatelského rozhraní. Tato vrstva bude tedy zastávat podobnou úlohu, jako vrstva *Controller* návrhového vzoru *MVC*, jehož modifikace je v klientské aplikaci použita.

2.6.2 Návrh vrstvy pro řízení periferií

Tato vrstva tedy bude sloužit jako prostředník mezi ovládanými periferiemi a zbytkem systému, a tak se pokusíme nadefinovat veřejné rozhraní pro komunikaci. Implementací návrhového vzoru *fasáda (ang. Facade)* docílíme jednotné podoby tohoto rozhraní pro jinak různorodé třídy ovládající hardwarové komponenty.

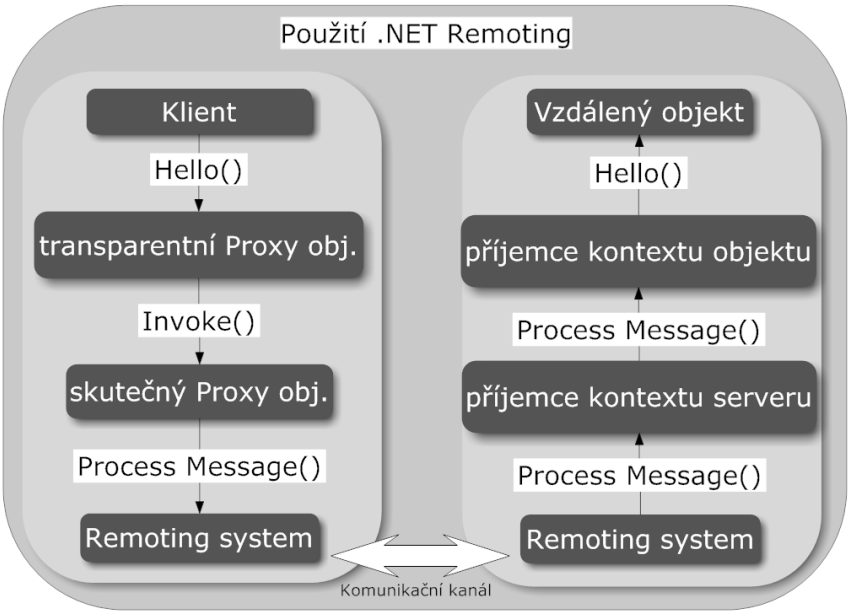
Použití služby (**Windows Service**) k hostování funkcionality namísto klasické (např. konzolové) aplikace je optimální volbou pro tuto část systému. Tento modul nevyžaduje žádné grafické rozhraní. Modul musí být přístupný ihned po nastartování serveru. Zde nám služba systému Windows velmi usnadní práci, protože naší službě stačí nastavit, aby se spouštěla automaticky při startu. Dalším přínosem je spolehlivost služby. Pokud by došlo k neočekávané vyjímce, služba je schopna se restartovat a během krátké doby se znovu spustit a to vše bez nutnosti zásahu uživatele. Jednou z posledních výhod je omezení neočekávaných situací zapříčiněných zásahem nezkušeného uživatele, které mohou eventuálně vést až k pádu systému, nemluvě o situaci, kdy uživatel tuto stěžejní službu v podobě klasické aplikace zapomene spustit. Ale nic není zadarmo a ani při použití *Windows Services* tomu není jinak.

Problém spočívá v komunikaci zbytku systému se službou. Pokud by tato služba nabízela své rozhraní přímo pro využití ze vzdáleného počítače (mobilního telefonu), bylo by nejvhodnější volbou použití služeb soketů TCP/IP. Server by v tomto případě musel vytvořit interní *Socket Server* (v jazyce C# TcpListener), který by naslouchal na veřejném portu. Pro každý příchozí požadavek by vytvořil vlastní vlákno, ve kterém by

se z příchozího proudu přečetl požadavek, který by byl následně předán ke zpracování naší službě. Po obsloužení požadavku by se odpověď zapsala do výstupního proudu.

Jelikož ale budou s touto službou komunikovat jen moduly běžící na lokální stanici (WCF Server, klientská aplikace pro dotykové LCD), je mnohem lepší alternativou využití technologie **.NET Remoting**.

2.6.2.1 .NET Remoting Podle knihy [2, kapitola 21] je technologie .NET Remoting intranetovou obdobou webových služeb, která však nemá zbytečnou režii v podobě serializace objektů pro přenos prostřednictvím protokolu SOAP. Každou součást .NET Remoting lze rozšířit, nebo dokonce nahradit vlastním řešením. Nejlépe tuto technologii vystihují dvě fráze: **Web Services Anywhere** (webové služby všude - tím je míněno, že službu lze využít v jakékoli aplikaci za použití jakéhokoli typu přenosu) a **CLR Object Remoting**(vzdálená komunikace společných objektů - zprostředkovává „proxy" instance objektu pro vzdálené volání spolu se všemi jeho vlastnostmi).



Obrázek 7: Struktura použití .NET Remoting

Architekturu .NET Remoting lze používat pro práci s objekty v jiné aplikační doméně bez ohledu na to, zda jsou komunikační systémy spuštěny ve stejném procesu, nebo dokonce ve stejném systému. Detailnější rozbor této zajímavé technologie je nad rámec této práce, proto vás tedy znovu odkazuji na výše zmíněnou publikaci, kde se dozvíte více.

Jak přesně začleníme tuto technologii do našeho vlastního kódu se dozvíte v sekci 3 zabývající se implementací. Z obrázku 7 je zřejmé, že klientská aplikace si vytvoří zástupný (Proxy) objekt skrze něj volá požadované metody. Tento proxy objekt spustí mechanismus, kde dojde nejprve k serializaci požadavku, dále přenesení požadavku na server po

zvoleném kominikačním kanále (TCP, UDP, HTTP, . . .). Na serveru je požadavek zpětně deserializován a po identifikaci příslušného vzdáleného objektu je mu požadavek předán ke zpracování. Odpověď prochází stejnou procedurou, pouze v opačném pořadí.

Implementací .NET Remoting zpřístupníme veřejné metody pro využití na lokální stanici. Tyto metody budou konzumovány klientskou aplikací pro dotykovou LCD obrazovku běžící na serveru.

2.6.3 Návrh realizace vstupně/výstupních bran pro řízení jednotlivých prvků inteligentní domácnosti

Právě nyní je vhodný okamžik pro návrh řešení, jak vůbec se budou fyzicky ovládat jednotlivé komponenty domácnosti. Z předchozí analýzy již známe přesné požadavky klienta. Víme, že bude potřeba zapnout/vypnout osvětlení, pořídit snímky z kamery, sepnout alarm apod. Pro budoucí použití tohoto projektu a další možné potřeby jiného klienta si musíme zvolit takové technologie, abychom se co možná nejvíce odprosili „jednoduchého" prvoplánového řešení, a tak si zajistit možnost snadné rozšířitelnosti o nové komponenty. Musíme tedy vymyslet, nebo lépe řečeno najít, řešení nabízející univerzální možnost ovládání rozličných periferií, které integrujeme do našeho systému. Je více než jasné, že k ovládání např. světel budeme muset použít digitální relé, které umožní sepnutí (rozepnutí) elektrického obvodu na základě příchozího signálu. Při průzkumu trhu (viz. kapitola 4.2 jsme zjistili, že existuje několik ovládacích prvků umožňujících „ovládat" právě spínání digitálních relé a schopných komunikovat po síti.

2.6.3.1 Ovládání periferií pomocí bezdrátových technologií

Než přejdeme k výběru vlastního zařízení pro ovládání, rozhodněme se nejprve, jaký bude optimální komunikační kanál pro dorozumívání se s touto deskou. Nejprve si prosím prohlédněte tabulku 2 srovnávající použitelné bezdrátové technologie.

Zigbee je bezdrátová komunikační technologie standardu IEEE 802.15.4. Podobně jako technologie Bluetooth, je technologie zigbee určena k propojování zařízení v sítích *PAN* (Personal Area Network). Hlavní doménou pro nasazení jsou aplikace s napájením na akumulátor, kdy při značně nižší spotřebě poskytuje výrazně delší dosah signálu v porovnání s technologií Bluetooth, ovšem na úkor horší přenosové rychlosti. Nižší přenosová rychlost zajišťuje vyšší odolnost proti rušení. Tento fakt předurčuje technologii ZigBee pro využití v průmyslových aplikacích, kde si vystačíme s rychlostmi v řádu několika desítek Kb/s.
Pro ZigBee je v Evropě vyhrazeno pásmo 868 MHz s jedním kanálem a přenosovou rychlostí 20 Kb/s. ZigBee síť se skládá ze tří druhů uzlů.

- **ZigBee coordinator** - v síti je vždy jeden. Toto zařízení se stará o chod sítě. Umí ukládat bezpečnostní klíče koncových uzlů. Synchronizace jednotlivých zařízení v síti ZigBee, potažmo koncových zařízení s koordinátorem sítě je realizována na základě takzvaného rámce *beacon*. Synchronizační autoritou je zde právě tento koordinátor sítě, který v daných okamžicích vysílá synchronizační sekvence(beacon). Sekvence přijímají ostatní zařízení a synchronizují

Technologie	GPRS/GSM	Wi-Fi 802.11b	Bluetooth 802.15.1	ZigBee 802.15.1
Aplikační zaměření	Široké oblasti (Hlas a Data)	Web, email, video	Náhrada za dat. kabel	Monitorování a řízení
Sys. zdroje (paměť)	16MB a více	1MB a více	250 KB a více	4-32KB
Životnost na baterii (dny)	1-7	0,5-5	1-7	100-1000
Přenosová rychlost (Kb/s)	64-128	11 000	720	20-250
Komunikační dosah (m)	1000 i více	1-100	1-10	1-100
Výhody	Dosažitelnost	Rychlost, flexibilita	Cena, jednoduchost	cena/výkon

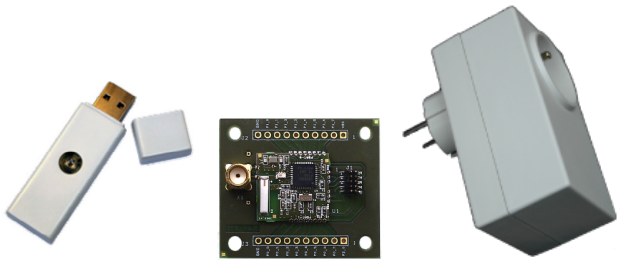
Tabulka 2: Srovnání parametrů bezdrátových technologií

se podle nich s vysílací stranou, tedy s koordinátorem. To nám umožňí koncová zařízení na dlouhou, předem definovanou dobu uspat, a značně tak snížit jejich spotřebu. Interval synchronizačních sekvencí může být nastaven v rozmezí 15 ms až po 15 minut. Pro přenos je pak možné využít tzv. super-rámce, začínající právě sekvencí beacon, po nichž následuje interval CFP(interval, v němž zařízení soutěží o přístup k médiu), kdy zařízení volně soutěží o přístup k médiu. Ten je případně následován intervalem s rezervovanými časovými sloty pro prioritní přenosy (tzv. GTS). Koordinátor zasílá v beacon sekvenci pro jednotlivá koncová zařízení také informace, zda jsou pro ně k dispozici data, či nikoliv. Pokud ano, koncová zařízení si je vyžádají a přijmou je v rezervovaných slotech. Pokud síť funguje bez sekvencí beacon, dotazují se jednotlivá zařízení periodicky koordinátora. Komunikace potom probíhá za absence vyhrazených slotů.

- **ZigBee router** - umožňuje vzájemnou komunikaci (výměnu dat) mezi ostatními prvky sítě.
- **ZigBee koncové zařízení** - jsou takové uzly sítě, které obecně z důvodu úspory energie mívají omezenou funkcionalitu(co se zigbee sítě týče). Umí komunikovat pouze s routery, nebo koordinátory. Většinu doby jsou ve stavu *čekání na signál*.

Z našeho embedded serveru bychom mohli snadno vytvořit ZigBee *koordinátor* za použití zařízení sloužícího k tvorbě vlastní ZigBee sítě **Ubee dongle** vyráběného společností *Cleode*. K tomu máme možnost pořídit si již hotová koncová zařízení (spínače zásuvek, teploměry, pohybová čidla apod.), nebo OEM verze desek umožňujících reagovat na příchozí události, nebo zasílat data (viz. obrázek 8). Použitím modulu pro zásuvku je možné ovládat téměř jakýkoli domácí spotřebič. Pro

naše potřeby si můžeme představit připojení kávovaru, který nám ráno uvaří čerstvou kávu, než vstaneme. K tomu využijeme plánovač, jehož návrh je součástí jiné diplomové práce ([8]). Podrobnější specifikace naleznete v příloze A.1.



Obrázek 8: ZigBee OEM řešení (Clode)

2.6.3.2 I/O Moduly s digitálními relé S ohledem na nižší finanční náročnost řešení jsme se rozhodli pro prvotní realizaci využít desek osazených digitálními relé prvky české firmy Papouch. Tato společnost má ve svém portfoliu řadu desek s Označením *Quido*, které se liší počtem vstupů/výstupů. Jednotlivé moduly se připojují přes sériový port RS232, sběrnici RS485, přes USB nebo Ethernet.

Výstupy jsou osazeny výkonovým relé s přepínacím kontaktem. Základní možností je kontakty relé okamžitě sepnout nebo rozepnout. Také je možné kontakty nastavit do požadovaného stavu na dobu určitou.

Vstupy jsou připraveny pro připojení kontaktu nebo napětí. Jsou dvoustavové - umí tedy rozlišit stavy připojeno/nepřipojeno napětí, respektive sepnutý/rozepnutý kontakt. U vstupů je možné zapnout funkci čítač a počítat impulzy externích čidel s impulzním výstupem (pohybová čidla - použitelná např. pro automatické rozsvícení světel). Stav vstupů je dostupný na dotaz z nadřazeného systému nebo lze nastavit mód, kdy se odešle automaticky zpráva o změně stavu vstupu.

Obrázek 9: Quido ETH 4 - deska s digitálními relé

Pro pokrytí potřeb našeho klienta postačuje verze *Quido ETH 4/4* viz. obrázek 9. K dispozici tedy budeme mít 4 vstupy a 4 výstupy. Tato deska tedy komunikuje prostřednictvím ethernetové LAN sítě (10/100MB). Tato volba byla podmíněná možností využití stávající lokální sítě u klienta v domě. Použití této verze má další bezespornou výhodu v podobě možnosti budoucího připojení další desky. To je přínosem zejména do budoucnosti, kdy může klient časem vznést nové požadavky na přidání dalších ovládaných elementů. Při implementaci tedy budeme počítat s touto možností současného zapojení více těchto zařízení a budeme systému muset pomocí konfiguračního souboru sdělit, s kolika Quidy pracuje. Systém pak sám podle této konfigurace nabídne v UI adekvátní ovládání na základě zvoleného profilu ovládané periferie.

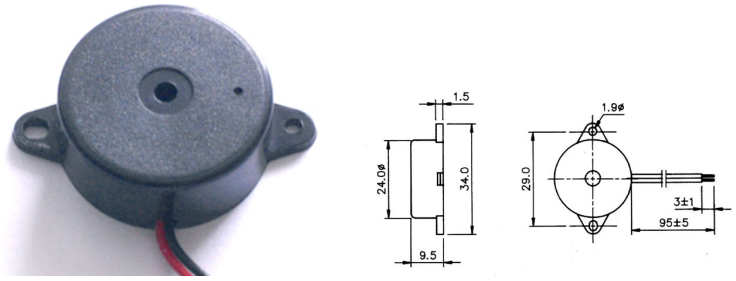
Všechny typy desek Quido lze ovládat podle dobře zdokumentovaného protokolu *Spinel*.

SPINEL - má jak svou textovou ASCII variantu, tak i binární podobu. Celou specifikaci protokolu si pak můžete prohlédnout v příloze A.2. Pro představu zde (obr. 10) uvedu jednoduchý příklad v textové podobě, který sepne relé číslo 2 na modulu s adresou 1.

Dotaz	Odpověď		Vysvětlení
*B1OS2H↵		*B	Prefix
		1	Adresa Jako adresu lze také použít znak $. Tento znak je univerzální adresou a funguje pokud je na lince jen jeden modul.
		OS	Kód instrukce pro změnu stavu výstupu
		2	Číslo výstupu
		H	Kód sepnutí (High)
		↵	Ukončovací znak (enter)
	*B10↵	*B	Prefix
		1	Adresa modulu
		0	Potvrzení
		↵	Ukončovací znak (enter)

Obrázek 10: Spinel - příkaz pro sepnutí relé

Pomocí Quida realizujeme ovládání bzučáku, který bude sloužit jako znamení pro přivolání pomoci (jiného člena domácnosti). Musíme si uvědomi, že Quido samo o sobě neprodukuje žádné výstupní napětí, ale chová se jako chytrý, dálkově ovládaný vypínač, který umožní sepnout, nebo rozpojit elektrický okruh. Pro ovládání bzučáku implementujeme několik druhů pulzních signálů. Tím ve výsledku umožníme různé zvukové signály, které mohou mít různé významy. Námi použitá sirénka (viz obr. 11) potřebuje napětí vstupní 3-30V DC (budeme jí tedy napájet pomocí baterií) a výstupním výkonem 85dB.



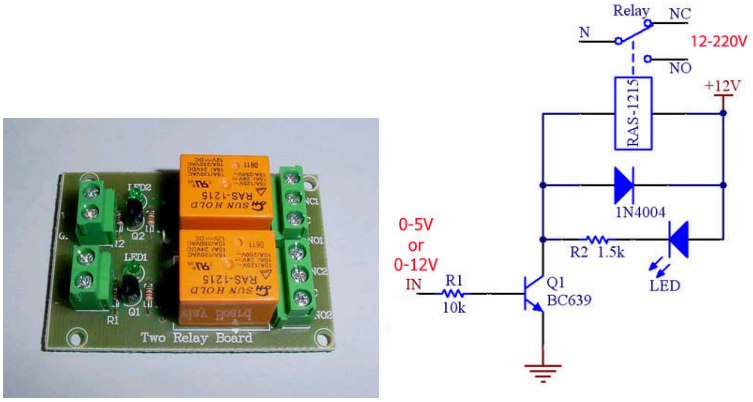
Obrázek 11: Bzučák pro přivolání pomoci

Dále pomocí Quida umožníme ovládání osvětlení. Technickou překážkou jsou limity pro maximální propustné napětí na spínaných portech, které činí 120V AC a 60V DC, které nám neumožní přímé zapojení Quida do domácí elektrické sítě. Pro ovládání osvětlení je nezbytné použít dalšího digitálního relé certifikovaného pro silnoproud (230V), které bude mít spínací napětí nejlépe 12V DC. Ve výsledné podobě tedy bude Quido spínat relé zapojené odborníkem do elektrického rozvodu v domácnosti. Pro velké množství ovládaných elektronických zařízení se vyplatí investovat např. do zařízení *S800L I/O* společnosti ABB (obrázek 12), které slouží jako ovladatelný spínací rozvaděč s dostatečným počtem portů pro pokrytí potřeb nejnáročnějšího zákazníka. Z finančních důvodů v prvotní fázi realizace použijeme vlastní spínací relé desky, která každá z nich umí sepnout a rozepnout 2 obvody (ovládat dvě světla). Jak je ze schématu na obrázku 13 patrné, použili jsme digitální relé (RAS-1215 se spínacím napětím 12V DC). Samotné vyrobení desky podle našeho návrhu jsme přenechali odborníkům, stejně jako následné zapojení do elektrické sítě.

Obrázek 12: Dálkově ovladatelný spínací rozvaděč S800L I/O

2.6.4 Návrh spojové vrstvy

Zbývá tedy ještě vhodným způsobem zpřístupnit rozhraní také pro potřeby mobilního klienta (dálkového ovládání), který bude muset pro volání použít síť. Konkrétně tedy domácí síť Wi-Fi (viz. návrh architektury). Možná se ptáte, proč nepoužít rozhraní .NET Remoting také pro potřeby dálkového ovládání? Je to především proto, že .NET Remoting mobilní telefon nemá prostředky pro tvorbu .NET Remoting klienta. Proto jsme se



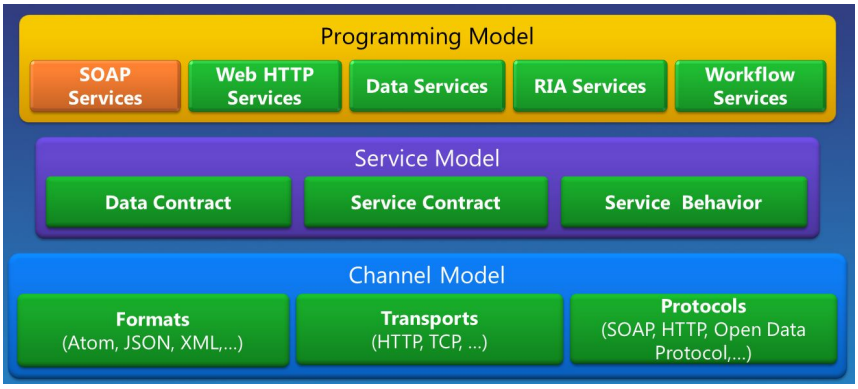
Obrázek 13: Naše deska pro ovládání světel

rozhodli pro zveřejnění rozhraní pomocí platformě nezávislých webových služeb (web services). Při testování systému jsme však narazili na výkonnostní problémy způsobené přílišnou režií. Jak je známo, webové služby jsou založeny na použití XML. Veškerá komunikace mezi serverem a klientem probíhá v textové formě pomocí SOAP protokolu. Veškeré požadavky a odpovědi se tak musí serializovat do podoby XML souboru.

2.6.4.1 Windows Comunication Foundation Webové služby jsme po další úvaze nahradili modernější technologií **WCF** (Windows Comunitation Foundation). WCF je součástí .NET Frameworku od verze 3.0. Servisně orientovaná architektura je vhodná pro tvorbu distribuovaných aplikací. WCF dokáže při správné konfiguraci plně nahradit starší webové služby, takže si stále zachovává platformní nezávislost. Ošidíme se ale tak o výhody, které tento framework přináší. Jak si můžete povšimnout na obrázku 14, WCF služby dokáží využít opravdu široké spektrum komunikačních kanálů počínaje protokolem webových služeb SOAP, přes zabezpečené HTTPS, až po TCP. Služby jsou typicky popsány pomocí **WSDL** (Web Services Description Language), což umožní automatizovanou tvorbu proxy tříd na straně klienta, který službu využívá. WCF má oproti Web Services nesporné výhody co se rychlosti týče. WCF využívá binární serializaci pro přenos dat. To si může dovolit díky tomu, že WCF má širokou základnu definovaných primitivních datových typů.

Dobrou zprávou je, že .NET Compact Framework (v 3.5) instalovaný v telefonech s Windows Phone 7 nativně podporuje tvorbu klientských aplikací konzumující WCF služby. WCF navíc nabízí několik způsobů pro zabezpečení služeb, a to jak v podobě šifrovaného komunikačního kanálu, tak i autentizaci a autorizaci požadavků. Existují tři možné způsoby, jak WCF služby hostovat:

1. WCF služba může být hostována v IIS (od verze 5.5, plnohodnotně však až od verze 7). Jde asi o nejběžnější a nejpraktičtější způsob. IIS7 nám dokáže zpřístupnit popis služeb ve formátu WSDL. IIS umí dále dobře pracovat s uživatelskými právy (umožňuje mimo jiné autentizaci a následnou autorizaci oproti uživatelským účtům



Obrázek 14: Architektura Windows Comunication Foundation (WCF)

systému Windows). Služba je navíc stále k online. Navíc budeme potřebovat IIS pro zpřístupnění webového rozhraní.

2. WCF služba může být hostována v klasické aplikaci (konzolová aplikace, WinForms aplikace, WPF aplikace). To pro naše účely není příliš vhodné, protože s ukončením aplikace se znepřístupní také služba.

3. WCF služba může být hostována ve klasické službě systému Windows (jde o tzv. *self-hosted* WCF Service). Toto je vhodná varianta, pokud není k dispozici IIS server. Při testování jsme zde však narazili na drobné implementační problémy.

2.7 Tvorba obrazu Windows Embedded

V první verzi řešení jsme jako operační systém embedded serveru zvolili vlastní image založený na Windows Embedded CE 6. WinCE je modulární, 32-bitový „real-time" operační systém určený pro malá zařízení, která si vystačí s 1MB paměti. Finální verze WinCE 6 R3 podporuje procesorové platformy ARMv5, MIPS, Intel x86 a SH4, dokáže využít až 4GB datové úložiště. Pro náš systém jsme potřebovali (mimo jiné) moduly pro práci s UI, .NET Framework (v základu je ve verzi 2.0, je tedy nutné jej poměrně náročným postupem upgradovat), moduly pro práci se sítí (Ethernetové ovladače, podpora síťových protokolů jako TCP/IP), IIS (v základu verze 5.5, kterou je možné povýšit maximálně na verzi 6). Službu IIS serveru potřebujeme pro hostování webových služeb a webového rozhraní pro ovládání.

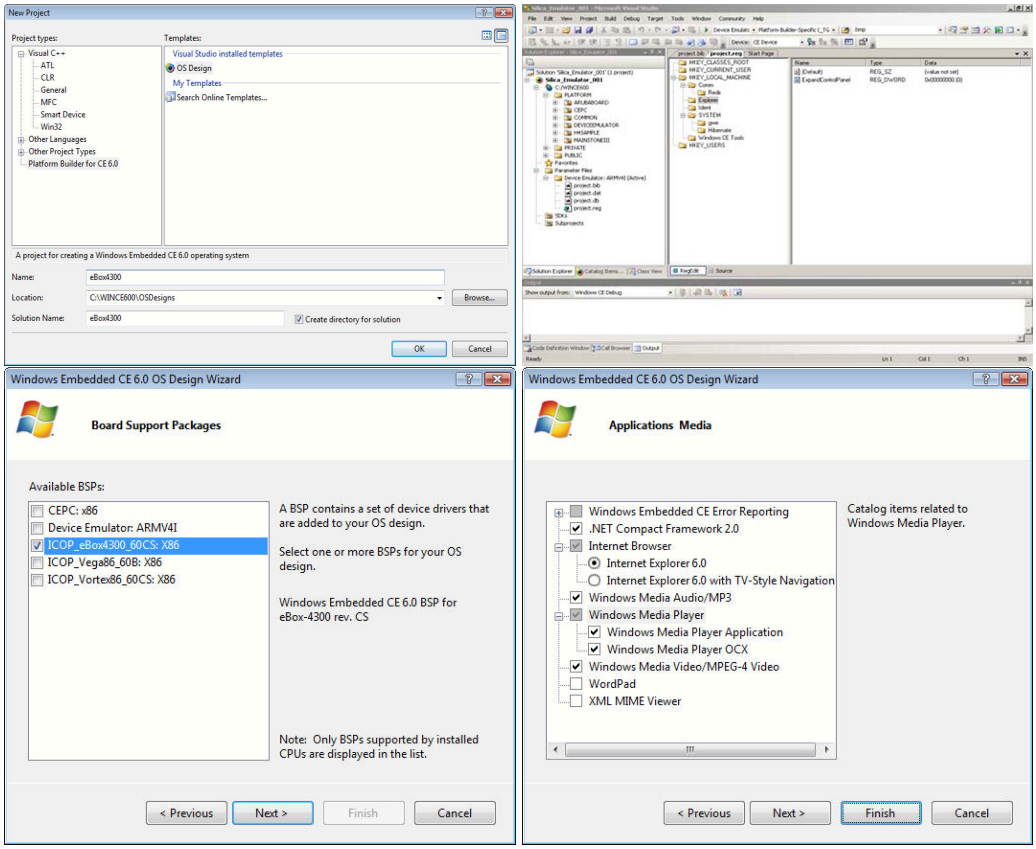
Než přistoupíme k samotné tvorbě našeho obrazu, vězte, že budeme potřebovat následující softwarové vybavení:

1. CD Windows CE 6 a další 2 CD s updaty na verzi CE6 R2 a následně CE6 R3
2. Visual Studion 2005 SP1 (pozor, nesmí být novější) a dále je potřeba doinstalovat *Platform Builder Ad-In*
3. Nainstalovaný operační systém Windows XP SP3 (opět není možno použít Windows Vista/7)

Pokud máme vše potřebné k dispozici, nainstalujeme si z CD Windows CE 6 komponenty pro tvorbu vlastních obrazů. Následně Provedeme aktualizace až na finální verzi WinCE CE6 R3. Obraz vytvoříme jako nový projekt typu Platform Builder ve Visual Studiu 2005, který po nás bude požadovat cestu k umístění zmíněných komponent.

Windows CE je specifickým softwarem z toho důvodu, že je dodáván v podobě zdrojových kódů (na rozdíl od Windows XP Embedded, který je dodáván jako balíček binárních knihoven), je tedy možné si v případě potřeby upravit dokonce jádro systému. Je tedy nejprve třeba, pomocí wizard nástroje **Platform builder**(v balíčku developers tools), vybrat cílovou hardwarovou platformu koncového zařízení(v našem případě x86 pro desku ALIX1D). Dále zvolíme připravený profil vytvořený na základě hardwarové konfigurace naší desky, který připraví nezbytný základ komponent pro běh systému. Dále si ručně přidáme naše zvolená rozšíření(IIS, .NET Framework, . . .), ovladače hardwarových komponent, jako drivery pro webovou kameru, wi-fi síťovou kartu a jiné. Pro ilustraci si můžete prohlédnout některé kroky na obrázku 16.

Po přidání veškerých potřebných modulů a kontroly závislosti všech komponent musíme náš projekt zkompilovat. Kompilace trvala něco přes dvě hodiny. Vliv na dobu kompilace mohla mít skutečnost, že jsme kompilaci prováděli ve virtuálním systému Windows XP. Výsledkem kompilace je hotový obraz systému, který je nutno překopírovat na Compact Flash kartu suplující pevný disk. Před tím je však třeba kartu připravit. Kartu je třeba naformátovat tak, aby používala souborový systém FAT16, dále je nezbytné nastavit jí bit označující vlastnost *BootFlag* z hodnoty 0 na BootFlag=1. Tím říkáme že paměťová karta slouží k zavádění operačního systému (podobně, jak je tomu u klasických pevných disků). Námi vytvořený obraz zabíral 490MB místa.



Obrázek 15: Tvorba obrazu Windows CE

2.7.1 Přechod na Windows Standard Embedded 7

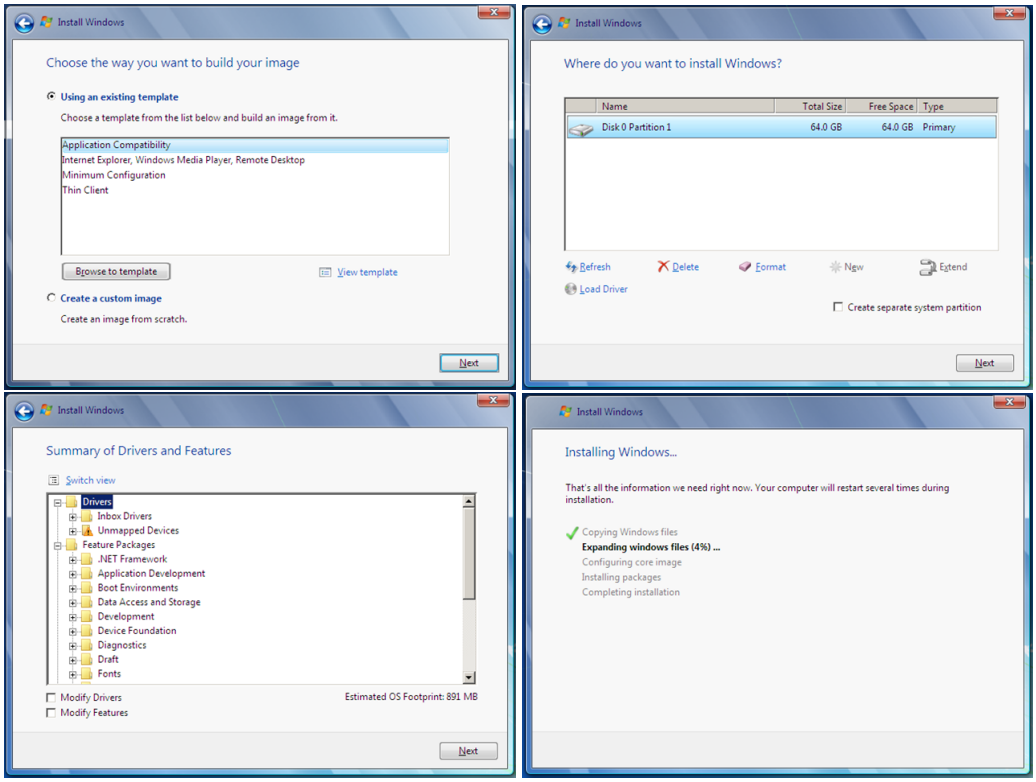
Po nasazení u klienta, kde se objevily jisté výkonnostní problémy jsme byli nuceni z Windows CE 6 přejít na nový operační systém. Změna šla ruku v ruce s výměnou desky ALIX1D, za desku novou. Díky dlouhé reakční době webových služeb, jsme byli nuceni přejít na použití WCF. WCF však potřebují pro svůj běh IIS ve verzi 7 (jinak se chovají stále jako klasické Web Services). Bohužel neexistuje způsob, jak na Windows CE provozovat IIS 7. Z toho vyplynula nutnost přechodu na **Windows Standard Embedded 7** (WES7). Pro otestování jsme použili 180-denní trial verzi volně dostupnou na stránkách společnosti Microsoft.

Obraz WES7 se tvoří odlišným způsobem, než je tomu v případě WinCE. Změna je také v tom, že již nemáme možnost přístupu k samotnému kódu operačního systému, což nám však nemusí zase tolik vadit. WES je možné nainstalovat několika způsoby.

1. Pokud je na cílovém zařízení provozován již nějaký OS, tak je možné opět pomocí nástrojů sady *developer tools* spustit službu **Target Designer**, která si „osahá" hardwarovou konfiguraci zařízení. Výsledkem je soubor, pomocí kterého je možné vytvořit základ obrazu, který je dále modifikovatelný pomocí designeru, obdobně jak tomu bylo u WinCE. Výhodou je zde, že obraz nevytváříme přímo na cílovém, méně výkonném zařízení. Zkompilovaný obraz opět nakopírujeme na použitou diskovou jednotku (CompactFlash, SSD, HDD).

2. Pohodlnějším způsobem instalace, je připojit k zařízení optickou mechaniku a spustit instalaci přímo z DVD. V první fázi se opět provede fáze mapování přítomného hardwaru. Po zjištění konfigurace se spustí wizard umožňující doinstalování požadovaných komponent (.NET Framework 4, chybějící ovladače, IIS 7.5, podpora síťových služeb, podpora GUI, podpora multimédií, ...). Posledním krokem je opět nezbytná kontrola kompatibility a závislostí zvolených součástí budoucího systému. Výsledný obraz se pak sám nakopíruje na zvolené datové úložiště.

Při všech možných postupech instalace jsme však narazili na problém. Instalátor vždy uvázl při inicializaci instalace v momentě kopírování dočasných souborů. Při snaze vyřešit tento problém jsme zjistili, že pro instalaci a běh systému potřebuje mít zařízení alespoň 512 MB paměti RAM. Naše deska ALIX1D má bohužel pouze 256MB a nenabízí slot pro osazení další pamětí. To je důvodem, proč bylo třeba pořídit i novou, výkonnější embedded desku. Jak však v kapitole 2.8 popisuji, cena nové výkonné desky, která by měla nahradit zapůjčenou desku ALIX1D je díky dnešním cenám velmi nízká. Naše finální verze zabírá 3,3GB diskového prostoru.



Obrázek 16: Tvorba obrazu WES7

2.8 Výběr vhodné platformy pro embedded zařízení

Nyní, když již máme alespoň základní představu o tom, jaké nefunkční požadavky jsou na systém kladeny, je na čase vybrat správnou platformu pro náš server a také pro mobilní zařízení. V této kapitole si porovnáme jednotlivé možnosti platforem použitelných pro naše účely.

Nejprve se tedy podíváme na jednotlivé, dnes běžně používané, platformy pro embedded zařízení. Při výběru však musíme myslet trošičku dopředu a mít na paměti, že toto zařízení bude muset být podporováno některou z distribucí operačního systému z rodiny Windows CE. Budou nás tedy zajímat zařízení využívající mikroprocesory typu ARM, MIPS a x86. Zároveň ale musíme dbát na to, abychom měli k dispozici dostatečný výkon a samozřejmě při tom se udržet v rozumné cenové relaci. Dále nás bude zajímat přítomnost komunikačních periferií. Komunikačními periferiemi jsou zde myšleny vstupně/výstupní brány, jako například paralelní port, sériový port, USB konektory, a dále síťová rozhraní jako Bluetooth, ethernetový port RJ-45, Wi-Fi apod.

2.8.1 ARM architektura

Na začátku bych si dovolil citovat slova ředitele ARMu Warrena Easta:

> Cílem architektury ARM není ovládnout PC trh. Tam je jasným vládcem Intel (tedy přesněji x86) a nemá valný smysl se do takového boje pouštět. ARM má svých odbytišť stále dost. Firmy jako Microsoft i Nvidia nám vlastně samy o sobě nacházejí či vytvářejí odbytiště nová, resp. jej směřují tam, kde sice stále vládne x86, ale není to pro tuto architekturu stěžejní jádro.

Procesory ARM jsou založeny na 32bitové **RISC** (Reduced Instruction Set - má redukovanou instrukční sadu a může tak mít více paralelních jednodušších instrukčních jednotek) architektuře. Výhoda ARMu spočívá především v jeho RISCové instrukční sadě - instrukce jsou vykonávány přímo hardwarem, nikoli mikrokódem. Fakt že procesor obsahuje menší počet hardwarových instrukcí dělá jeho návrh do jisté míry jednodušší. Vlastní procesor je menší, a tak má menší spotřebu, což se projeví na ceně za jeho provoz, protože se počítá se stálým během serveru. Jednodušší instrukční soubor umožňuje také snadnější vytvoření kompilátoru. Díky některým vlastnostem ARM je omezen i problém předpovídání skoků. Za výhody RISCových procesorů se považuje i větší počet univerzálně použitelných registrů. ARM má navíc některé vymoženosti jako jsou podmíněné instrukce.

Nízká spotřeba energie při vysokém výpočetním výkonu má zásadní význam hlavně v zařízeních napájených bateriemi. Proto dnes tyto procesory ovládají trh s malými přenosnými zařízeními, jako jsou multimediální přehrávače, mobilní telefony, kapesní počítače, kalkulátory atd.

V současné době jsou na trhu k dispozici procesory ARM ve verzích **ARM11 (založený na ARMv6xx)** a posledním vývojovým stupněm je **Cortex (založený na ARMv7xx)**. ARM11, který nabízí výkon 533MHz-1GHz, je hojně nasazován v dnešních mobilních telefonech. Pro představu uvedu několik příkladů. Nejznámějším představitelem je iPhone 3, HTC WildFirea a Nokie řad Nokie Nxx a Nokie Exx.

Procesor	Samsung Cortex-A8 ARM S5PV210AH-A0, 1GHz
Podporované OS	WinCE6.0, Android2.1, Android2.2
Operační paměť	Samsung K4T1G108, 1GB DDR2
Disková úložiště	2x SD Slot (každý až 32GB), 1x T-Flash Slot, 1x ATA konektor
USB konektory	4x USB 2.0
Možnost připojení dotykové obrazovky	Ano (50-pinový konektor pro až 12.1" LCD)
Video výstupy	VGA (1024x768), HDMI (1080p@30fps), DVI, TV-Out
Zvukový Vstup/Výstup	Ano/Ano
Síťová rozhraní	1x Ethernet RJ-45, 1x SDIO Wi-Fi (802.11 b/g/n)
Konektivita	2x RS485 sériová linka (COM5, COM6), 4x sériová linka RS232 s DB9 portem
Rozměry	320mm $\times$ 250mm
Cena	asi 15000 Kč

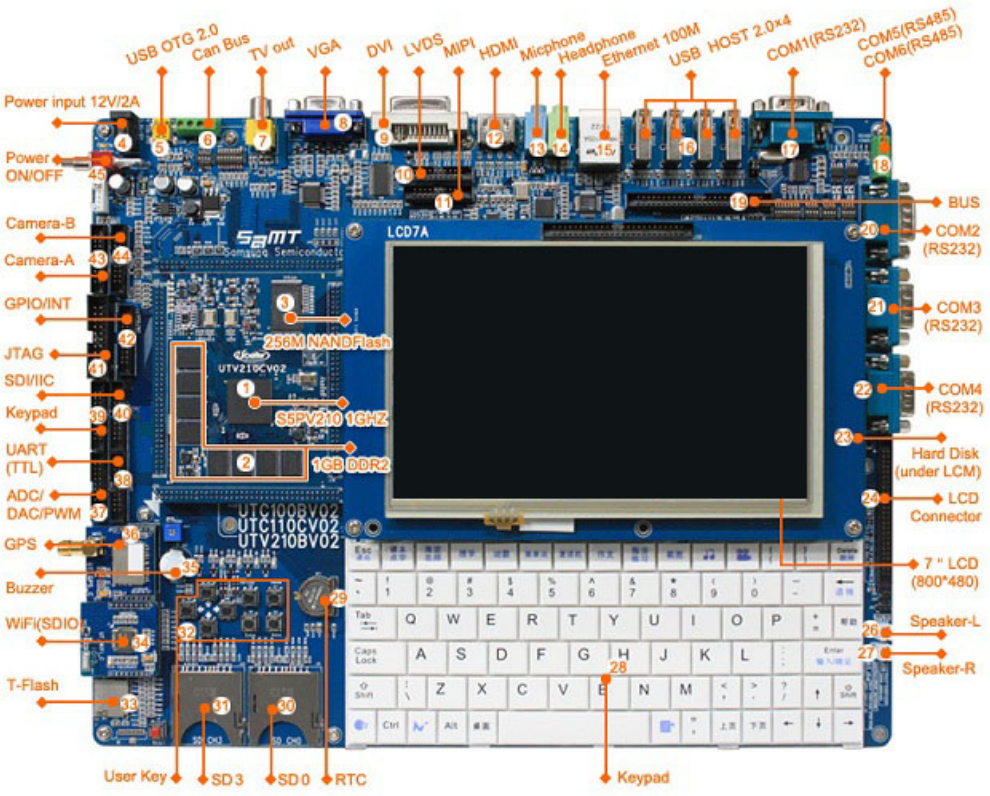
Tabulka 3: Specifikace ARM Kitu Samsung (S3C6410)

Cortex se naopak snaží prosadit na trhu s netbooky a herními konzolemi. A chce tak konkurovat procesorům Atom firmy Intel. Cortex procesory zaručují plnou kompatibilitu s technologiemi používaných v moderních aplikacích, jako jsou J2SE, Java (FX), Adobe Flash etc. Dnes tvoří základ pro většinu zařízení s Android OS postaveném na linuxovém jádře a hojně využívající právě technologií JAVA. Nabízí podporu zpracování jak 2D, tak 3D grafiky, a tak i možnosti přívětivého uživatelského prostředí, podporu multimediálních formátů, jako MPEG-4. Výkon se pohybuje v rozmezí 1GHz - 2,5GHz a je dostupný jak v jedno-jádrovém, tak až ve čtyř-jádrovém provedení se schopností namapovat až 4GB operační paměti a 1TB diskového prostoru.

Pro naše potřeby by byl vhodný např. model mini-ATX desky *S5PV210 ARM Development Kit* od společnosti *Samsung*. Jak plyne z jeho specifikace, kterou si můžete prohlédnout v tabulce 3, tak toto řešení skýtá dostatečný (až nadstandardní) výkon, spoustu vstupně/výstupních periferií a co se týče síťové konektivity, tak ani zde nic nechybí. Vadou na kráse je zde ale poněkud vyšší cena, která však za takto dobře vybavený KIT není nijak přehnaná. V uvedené částce bohužel nejsou zahrnuty jak třeba diskové jednotky, tak ani zobrazovací zařízení (dotyková LCD obrazovka), které opět toto řešení prodraží. Jak deska vypadá je zobrazeno na obrázku 17.

2.8.2 x86 architektura

Architektura x86 současně ovládá trhy se stolními počítači, přenosnými počítači a „malými" servery. V úvodu bych zmínil dnešní trend. ARM a x86 se pomalu začínají sbližovat. Zatímco x86 vychází z výkonných jader, která jsou postupně stále více optimalizována z hlediska spotřeby, ARM má kořeny v super úsporné architektuře, které je dodáván stále



Obrázek 17: Deska ARM Kitu Samsung (S3C6410)

vyšší a vyšší výkon. Proto pomalu začínají mít uplatnění ve stejných aplikacích (např. netbooky).

Dříve jsme zde nacházeli převážně procesory s 32-bitovou instrukční sadou, dnes se ale setkáváme již s procesory 64-bitovými. Tyto procesory jsou opět postaveny na RISC architektuře, stejně jako drtivá většina moderních procesorů. Mnozí by mohli namítat, že se procesory této rodiny příliš nehodí do řekněme úsporného embedded zařízení, a to hlavně kvůli spotřebě a množství vyprodukovaného tepla při zatížení, které je třeba odvádět aktivními prvky, jako je např. ventilátor. Až do nedávné doby by měli tito lidé pravdu. Revoluci zde však přinesla firma Intel se svými procesory Atom, které poprvé spatřily světlo světa začátkem roku 2008.

Intel Atom je značka procesorů patřící do rodiny x86 a x86-64. Jsou konstruované speciálně pro zařízení vyžadující velmi nízký odběr energie a přitom stále vysoký výkon (hlavně netbooky, PDA, smartphony či ultra-mobilní PC). Procesor je vyráběn 45 nm technologií. V jádře je hardwarová podpora pro vykreslování 2D/3D grafiky, která si hravě poradí např. s MPEG-4, WMV9, H.264 a to i ve Full-HD (1080p @30fps) rozlišení.

V současné době se dělí do dvou vývojových větví, podle segmentu zařízení, pro která jsou určeny. V mobilnějších zařízeních, kde je hlavní důraz kladen na nízkou spotřebu (delší výdrž zařízení v chodu na baterii) se setkáme s procesory pod kódovým označením **Intel Atom Lincroft**. Procesor je tvořen jedním jádrem. Spadají zde procesory s označením Atom Z6xx. Takt procesoru se podle modelu pohybuje od 1.2GHz-1.9Ghz. Šířka sběrnice je v rozmezí 400-533MHz. Cahce paměť v jádru má hodnotu 512 KB a to vše při velmi nízké hodnotě TDP (Thermal Design Power) $<$ 2,5 W.

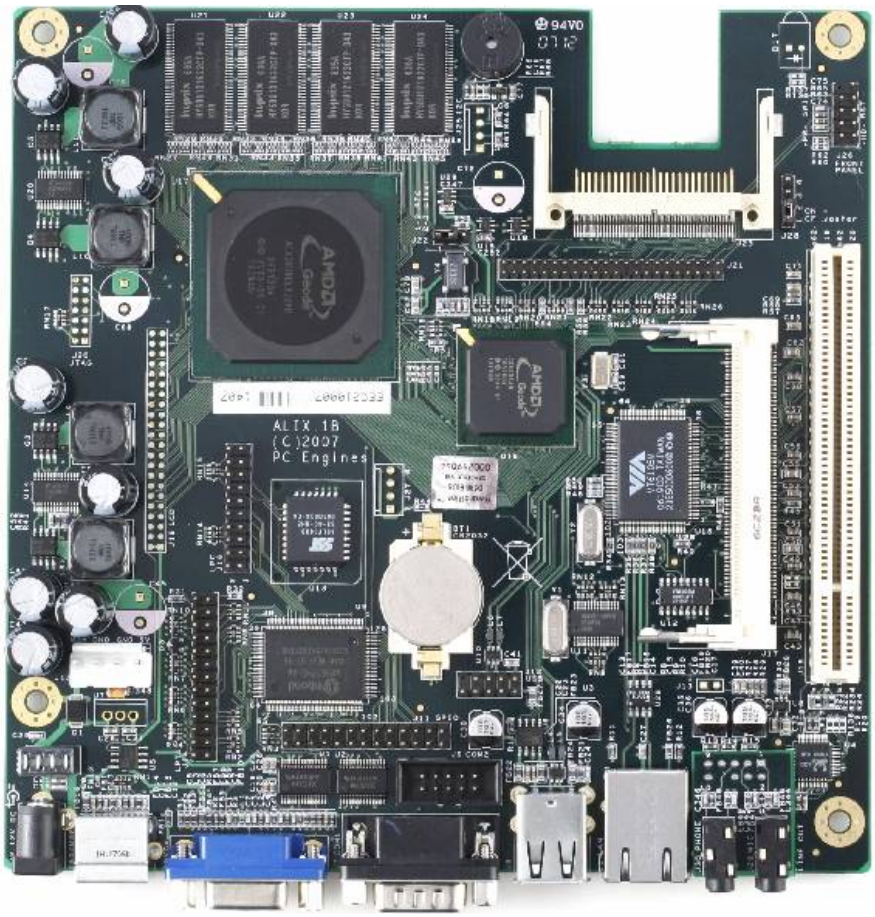
Do druhé kategorie, pod značením **Intel Atom Pineview**, se řadí procesory se značením N4xx a D4xx. Jde o jedno-jádrové procesory, kde kategorie N představuje úspornější, ale méně výkonnou variantu. Pak se zde dokonce setkáme i s prvními dvou-jádrovými zástupci N5xx a D5xx (N opět značí úspornější, ale méně výkonnou variantu). TDP je zde výrazně vyšší a to 6,5W až 13W, což nám může napovídat, pro jaká zařízení jsou určeny. Takt jednoho jádra je zde až úctyhodných 1,83GHz při podporované šířce sběrnice 667MHz a solidní cache pamětí o velikosti až 1MB.

Je tedy opravdu z čeho vybírat. Jednou z možných variant pro naše účely může být například mini-ATX deska *ALIX-1D*, která je zobrazena na obrázku 18. Podrobnější představu o hardwarové konfiguraci si můžete udělat z tabulky 4. Za poměrně nízkou cenu zde nalezneme v podstatě vše, co budeme pro naši inteligentní domácnost potřebovat. Deska nabízí dostatek komunikačních bran především v podobě USB portů. O potřebnou komunikaci v síti se zde postará Ethernetový konektor. Co by zde naopak mohlo vadit je skromný výkon, který daň za nižší cenu a spotřebu zařízení a také musíme brát v úvahu, že deska je na trhu již od roku 2007. Samozřejmě je třeba počítat s dalšími náklady za pořízení např. CF-karty, dotykového LCD panelu aj.

Zajímavější volbou by tak mohla být jedna z nejnovějších embedded desek firmy Jetway. Z široké škály výrobků se pro nás nejlépe hodí deska **Jetway NF96**, ve které se snoubí vysoký výkon (viz. specifikace v tabulce 5), moderní výrobní technologie. Jako krok správným směrem je přítomnost 2 slotů pro DDR2 paměti. Navzdory vysokému výkonu si zde, a to především díky modernímu procesoru Intel Atom D525 s maximální

Procesor	500 MHz AMD Geode LX
Podporované OS	WinCE6.0
Operační paměť	256MB DDR
Disková úložiště	1x CompactFlash slot, 1x 44 pin ATA konektor
USB konektory	4x USB 2.0
Možnost připojení dotykové obrazovky	„Ano" (USB + VGA)
Video výstupy	VGA (1024x768)
Zvukový Vstup/Výstup	Ano/Ano
Síťová rozhraní	1x Ethernet RJ-45, 1x (Mini PCI) Wi-Fi (802.11 b/g)
Konektivita	2x RS485 sériová linka (COM5, COM6)
Rozměry	185mm × 185mm
Cena	3000 Kč

Tabulka 4: Specifikace embedded desky ALIX-1D



Obrázek 18: Deska ALIX-1D

Procesor	Intel Atom D525 (2x 1.8 GHz, 1Mb cache)
Podporované OS	Windows Embedded 7
Operační paměť	2x DDR2 800/667 SDRAM (až 4GB) - neosazeno
Disková úložiště	2x konektor Serial ATA2 3 Gb/s - neosazeno
USB konektory	5x USB 2.0/1.1
Možnost připojení dotykové obrazovky	„Ano" (USB + VGA)
Video výstupy	VGA
Zvukový Vstup/Výstup	Ano/Ano
Síťová rozhraní	1x Gigabit Ethernet LAN RJ-45, 1x (Mini PCI) Wi-Fi (802.11 b/g/n)
Konektivita	1x RS-422/485 sériová linka COM1, 1x paralelní port, 1x LVDS
Rozměry	170×170mm
Cena	2300 Kč

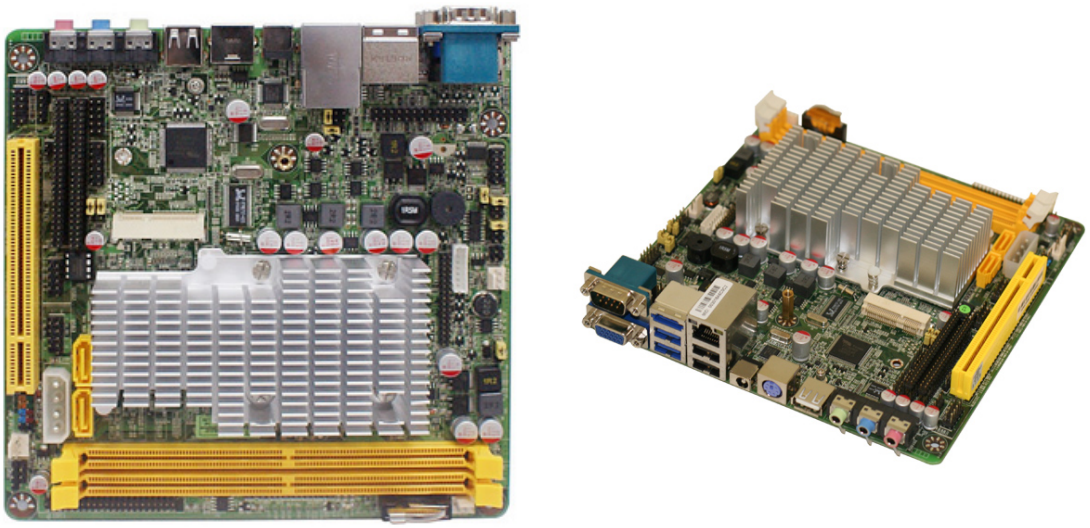
Tabulka 5: Specifikace embedded desky Jetway NF96

TDP do 13W, vystačíme s pasivním chlazením. Jak je patrné z obrázků 19, tak je zde přítomný dostatečný počet USB portů a ethernetový konektor pro připojení k síti. Postrádat zde ale můžeme např. slot pro Compact Flash paměťovou kartu, namísto které zde bude třeba pořídit např USB Flash disk, nebo lépe SSD disk (pro SATA konektor), který ovšem zařízení o zhruba 1500 Kč prodraží. Když k ceně ještě připočteme zhruba 600 Kč za 2GB operační paměti, tak se dostaneme na velmi přijatelnou cenu 4400 Kč za tichý a výkonný embedded server.

2.8.3 MIPS architektura

Poslední architekturou, která je pro tento projekt použitelná je označována jako MIPS, což je akronym anglického *Microprocessor without Interlocked Pipeline Stages*. Jedná se tedy o procesory bez automaticky organizované pipeline. To pro nás znamená, že procesor sice může zpracovávat více instrukcí současně (pouze za podmínky, že tyto instrukce nepoužívají stejné prostředky procesoru), ale o řazení instrukcí do pipeline se musí starat programátor, nebo překladač (nikoli samotný procesor, jak je tomu u procesorů s interlocked pipeline). Architektura MIPS opět spadá do kategorie RISC procesorů. Pro nás jsou v této kategorii zajímavé verze *MIPS32*® a *MIPS64*® architektury, které jsou cíleny právě na segment embedded zařízení.

MIPS32 architektura využíva load/store datový model, což znamená, že běžné instrukce nemohou používat operandy z paměti, ale pouze z registrů procesoru. Pro účely přenosu dat mezi registry procesoru a pamětí slouží právě speciální instrukce *load* a *store*. Ale to už zabíháme do přílišných detailů, které jsou nad rámec této práce. Co je však pro nás naopak může být zajímavé, je podpora i pro *real-time* operační systémy, mezi které se dá zařadit i Windows CE. Na čipu je ukryta i velmi dobrá podpora pro rychlé zpracování



Obrázek 19: Embedded deska Jetway NF96

grafických úloh, proto nikoho nepřekvapí, že se procesor prosadil v herních konzolích, jako např. Sony Playstation 2, nebo v set-top boxech, video kamerách apod.

Protože bychom v našem projektu nebyli plně schopni využít veškerý nabízený potenciál této architektury, tak se při výběru zaměříme spíše na předešlé dvě možnosti.

2.8.4 Volba platformy embedded zařízení

Musíme tedy vybrat ze dvou kandidátů. Obě možné varianty zcela pokrývají naše požadavky. Hlavním argumentem, proč jsme nakonec sáhli po platformě x86 (Intel Atom), je lepší dostupnost desek v příznivé cenové relaci. Dále je Intel Atom plně podporován operačním systémem Windows Embedded 7 ve verzi Standard. Platforma procesoru ARM najde i tak v našem projektu uplatnění a sice v mobilním zařízení, ale to trošku předbíhám. Dovolte mi tedy, abych ještě jednou zdůraznil, které konkrétní zařízení budeme chtít využívat pro nasazení konečné aplikace. Bude to deska **Jetway NF96** s procesorem Atom D525 s pořizovací cenou 5000 Kč (včetně pamětí, disku a boxu).

2.9 Výběr vhodné platformy pro mobilní zařízení

Při výběru mobilního zařízení musíme brát v úvahu opět několik faktorů. Zařízení bude muset být schopno komunikovat v lokální síti a to nejlépe přes Wi-Fi síť, neboť jak jsem již dříve v této kapitole zmínil, ostatní bezdrátové technologie nejsou pro nás v tuto chvíli příliš vhodné. Zařízení by mělo dále nabízet jistý komfort pro koncového uživatele, co se použitelnosti týče. Použitelností zde myslím především velikost a rozlišení displaye, protože stále bychom neměli zapomínat, pro koho je projekt tvořen. V dnešní době se již asi nemá příliš cenu zaobírat kapesními počítači (PDA), které byly nahrazeny chytrými

telefony (MDA), takže když se bavíme o mobilním zařízení, budeme tím mínit mobilní telefony. V úvahu také přichází ještě tablety, které nabízí vetší velikosti obrazovek, ale tuto možnost si ponecháme pro možnou budoucí modifikaci za účelem prodeje jinému klientovi. Mobilní telefon navíc poslouží klientovi i k telefonování. Proto jej nákup telefonu nebude tolik „mrzet" - neutratí peníze za zařízení, které by měl účelově jen pro ovládání inteligentní domácnosti.

2.9.1 Dostupné platformy

Než se zaměříme na výběr konkrétního telefonu na základě hardwarové specifikace, podívejme se nejprve v krátkosti na operační systémy, se kterými se můžeme setkat na dnešních mobilních platformách. Nebudu zde uvádět konkrétní přesná čísla, jak jsou jednotlivé OS zastoupeny, ale věžte, že přes 50% dnešních telefonů využíva již mrtvou platformu *Symbian*. Je tomu tak především z historických důvodů, protože Symbian byl spjat s telefony Nokia a Sony Erricson. Firma Nokia však nedávno oznámila, že končí s vývojem, dnes již nemoderního a těžkopádného OS Symbian, a do svých novývh telefonů bude nasazovat Windows Phone 7.

Druhým největším hráčem na poli mobilních OS je *iOS* společnosti Apple, s jejímiž výrobky je pevně spjat. Systém iOS tedy nenajdeme v jiných zařízeních, než v Apple iPhone a nově také v tablet-PC Aplle iPad. Byď jsou telefony iPhone výkonnými zařízeními s kvalitním a velkým displayem, je zde překážka pro vývojáře, kteří jsou nuceni si pro vývoj pořídit ne zrovna levné PC od Apple s operačním systémem Mac OS. Dalším problémem je transport aplikace do telefonu, pro který je třeba využít market, kde je třeba se za poplatek registrovat.

Největším dravcem je dnes asi *Android* OS společnosti Google, kterému je prorokováno, že brzy sesadí z trůnu právě stařičký Symbian. V jeho prospěch hovoří především otevřenost platformy založené na Linux jádře. Dnes mu patří asi 15% trhu, ale toto číslo neustále roste. Nově se začíná objevovat nejen v tabletech, ale také v netboocích. Rozhodně zde nenarazíme na problém s vývojem a následným nasazením aplikace do telefonu. Ještě bych zmínil, že pro vývoj aplikací je zde použita technolohgie JAVA.

Na posledním místě bych zde uvedl ještě Windows Mobile 6.5, který je již dnes také odsouzen k zániku, protože Microsoft vydal novou verzi Windows Phone 7 (WP7), která se snaží od této vývojové větve „distancovat". Windows Phone 7 zaujme především svým zcela přepracovaným vzhledem, ale také funkčním jádrem. Snaží se vydobýt si zpět dávno ztracené postavení a zapadnout mezi dnešní leadery iOS a Android. Pro vývoj je zde také třeba vlastnit PC s operačním systémem Windows, ale vývojové nástroje a SDK jsou zdarma. Problém je zde opět s nasazením hotové aplikace do mobilního zařízení, k čemuž je zapotřebí marketu. Aplikace je však možno nasadit na koncový telefon přímo z vývojového prostředí. Pro vývoj aplikací je požit .NET Compact Framework.

2.9.2 Volba mobilní platformy

Jak již bylo řečeno, končící platforma Symbian nepřichází v úvahu. Proto ve hře zůstávají platformy Android a Windows Mobile. Pro Android hovoří především otevřenost a kom-

patibilita s novým hardwarem. Hlavním jazýčkem na misce vah je však skutečnost, že embedded server bude využívat OS rodiny Windows využívající platformu .NET. Bylo by tedy hloupé se ošidit o možnost využití technologií specifických právě pro platformu .NET. Tím nechci říkat, že by bylo nemožné postavit řešení na konkurenční technologii JAVA, nebo dokonce současné použití obou těchto technologií. Ale zadělali bychom si zbytečně na další problémy.

Rozhodli jsme se tedy pro zachování konzistentních technologií firmy Microsoft. Z předchozí kapitoly 2.9 je zjevné, že nesáhneme po OS Windows Mobile 6.5, který se především díky nutnosti požívaní stylusu nehodí pro pohodlné ovládání. Ve prospěch nové „verze" (opakuji, nejedná se o novou verzi ve smyslu vylepšení verze stávající, ale jde o zcela nový OS) Windows Phone 7 mluví následující fakta:

- WP7 podmiňuje své nasazení minimálními hardwarovými nároky na použité mobilní zařízení. Tato nároky diktují zařízení mít alespoň 512 MB operační paměti, procesor s taktem minimálně 1 GHz a pro zajištění dobré uživatelské zkušenosti s ovládáním vyžaduje minimální rozlišení displaye na 480x800 px. Na rozdíl od např. od Android OS je striktně vyžadováno i použití hardwarových tlačítek pro funkce *zpět*, *start* a *vyhledat*.

- WP7 je tedy nasazován na poměrně výkonnostně dobře vybavená zařízení. Proto se při vývoji setkáme s podporou moderních technologií platformy .NET 4. Pro tvorbu uživatelského rozhraní UI je podle druhu vyvíjené aplikace možno použít buď Silverlight (pro business aplikace), nebo XNA (pro hry, nebo graficky náročné 3D aplikace). K tvorbě aplikací je zdarma dostupný vývojový balíček (Visual Studio 2010 Express a WP7 SDK), který s sebou přináší i povedený emulátor, který na rozdíl od emulátorů dřívějších nemá problémy při práci v síti.

Abychom jen nechválili Wondows Phone je poměrně mladou platformou a proto je dnes stále ještě obtížné v ČR sehnat zcela legální cestou telefon s tímto OS. Je to dle mého názoru zapříčiněno především absencí české jazykové mutace. Microsoft si je však tohoto neduhu vědom a v nedávné době oznámil, že se na ostatní jazykové mutace můžeme těšit při dalším updatu, který přinese mimo jiné také podporu Silverlight 4. U nás se zatím prodává asi jen 5 telefonů běžících na WP7. Z této chudé nabídky vyčnívá např. **HTC 7 Mozart**, jehož specifikaci si opět můžeme podrobně prohlédnout v tabulce 6, který oplývá kvalitním displayem, dobrým výkonem, výbornou konektivitou a to vše za cenu necelých 8000 Kč. Že jsou přítomna i zmíněná hardwarová tlačítka se můžeme přesvědčit z obrázku 20. Co by se dalo vytknout je použití staršího modelu procesoru, který je vyráběn ještě 65nm technologií. Nová verze ARMv7 Scorpio vyráběná novější 45nm technologií prodlouží výdrž zařízení při chodu na baterii až o 30%.

Procesor	Qualcomm QSD8250 (1GHz) - ARMv7 Scorpio (65nm)
Display	velikost 3,7″ (48 × 81 mm), rozlišení 480 × 800
Operační paměť	576 MB
Disková úložiště	vestavěno 8 000 MB
USB konektory	1x microUSB
Síťová rozhraní	Wi-Fi (802.11 b/g/n), Bluetooth (A2DP)
Výdrž baterie	360 hod. v pohotovostním režimu
Rozměry	119 × 60 × 12 mm
Cena	7600 Kč

Tabulka 6: Specifikace telefonu HTC 7 Mozart

Obrázek 20: Mobilní telefon HTC 7 Mozart

3 Implementace

V části implementace se nebudeme pokoušet detailně popisovat veškerou funkcionalitu. Zaměříme se zde pouze na některé klíčové segmenty. Po přečtení této kapitoly byste měli být schopni se orientovat ve zdrojovém kódu aplikací. Záměrně říkám aplikací, protože jsme striktně dodrželi plánovanou vícevrstvou architekturu. Při implementaci jsme navíc využili řadu návrhových vzorů a tzv. *best practicies*. V případě zájmu si můžete prohlédnout jednotlivé diagramy tříd v příloze A.3.

3.1 Implementace vícevrstvé architektury

Při návrhu celkové architektury naší aplikace jsme se inspirovali hojně používaným návrhovým MVC. Jistě jej všicni velice dobře známe, ale jen pro úplnost uvedu základní myšlenku. MVC, neboli Model, View a Controller jsou 3 základní vrstvy aplikace. Každá z nich se může dále dělit na vrstvy dílčí.

Model vrstva reprezentuje vrstvu persistentních dat. V aplikacích slouží objekty této třídy jako nositelé dat, které mají jen omezenou logiku. V naší konkrétní implementaci jsou jako model považovány objekty reprezentující vždy jednu konkrétní ovládanou periferii (např. světlo).

View vrstva obsahuje třídy, které slouží **výhradně** ke grafické reprezentaci získaných dat v podobě objektů tříd nacházejících se ve vrstvě modelu. View vrstva dále informuje vrstvu controller o událostech vzniklých interakcí s uživatelem. V našem případě je separace vrstvy UI nutná, pro budoucí tvorbu nového vzhledu pro potřeby klienta s jiným druhem postižení. V mobilní aplikaci reprezentují tuto vrstvu XAML soubory, které tvoří zobrazovací šablony pro Silverlight.

Controller vrstva reprezentuje aplikační logiku systému. Stará se o získávání dat, které formou objektů předává vhodnému zobrazení (View). Dále zpracovává události vzniklé ve vrstvě UI.

3.1.1 Implementace Windows Service

Jak již bylo řečeno v návrhu, Windows Service jsme zvolili především kvůli spolehlivosti, nižším nárokům(nepotřebuje GUI) a schopnosti zotavit se při chybě. Pro zaznamenávání chyb a vůbec celkového využití jednotlivých metod aplikace jsme implementovali jednotný systém pro logování chyb a událostí. Ze všech modulů aplikace je dostupná třída **Logger**, která nabízí statické metody pro záznam událostí. Záznamy jsou jednotně ukládány do logu systému, kde se dají pohodlně třídit a vyhledávat.

Samotná služba **Remoted Server Service** slouží k hostování a spouštění jednotlivých vláken, které každé z nich reprezentuje modul(controller) pro ovládání periferie, nebo skupiny periferií. Další důležitou úlohou je spuštění *TcpServerChannel* který umožní registrace kanálů jednotlivých controllerů pro použití metod vzdáleného volání skrze *.NET Remoting*. Služba tedy nabízí jednotné veřejné rozhraní pro ovládání rozličných („různě implementovaných") tříd, což je podstatou návrhového vzoru *Fasáda*.

Pro jednotný způsob spuštění výkonného vlákna implementují všechny řídící třídy rozhraní *IWinService*, které definuje metody *Start()* a *Stop()*, volané při spouštění respektive ukončení služby Windows Service. Tím zajistíme, že služba bude dostupná již při startu serveru a že se korektně ukončí při jeho vypnutí.

3.1.2 Implementace .NET Remoting

Zde si na krátké ukázce kódu ukážeme, jak zaregistrovat třídu pro ovládání skrze .NET Remoting a jak tento vzdálený objekt. Třída **CameraControllerProxy** reprezentuje vlastní objekt, který bude při spuštění služby registrován pro vzdálené volání. Tato třída obsahuje instanci třídy **WebCam**, která zprostředkovává vlastní ovládání kamery. CameraControllerProxy pak nabízí veřejnou propery, která umožní přístup k datům vráceným kamerou. O registraci se stará dceřiná třída CameraController, která navíc implementuje rozhraní potřebné ke spuštění controlleru v naší Windows Service. Při spuštění služby dojde k vlastní registraci objektu CameraControllerProxy.

```
public class CameraControllerProxy : MarshalByRefObject, IDisposable
{
    private byte[] _imgBytes;
    private Thread _imgSavingThread;
    private readonly int _imageCapturingDelayMilis;

    public byte[] ImageBytes
    {  get { return GetImageBytes(PathToCamImg); } }

    public CameraControllerProxy()
    {
        _imageCapturingDelayMilis = 2500;
        _runCapturing = true;
        IniWebCamCapture();
    }

    private void IniWebCamCapture()
    {
        _imgSavingThread = new Thread(() => DoCapturingImages(_imageCapturingDelayMilis));
        _imgSavingThread.SetApartmentState(ApartmentState.STA);
        _imgSavingThread.IsBackground = true;
    }
}

public class CameraController : CameraControllerProxy, IWinService
{
    public void Run()
    {
        TcpServerChannel _tcpServerChannel = new TcpServerChannel(8086);
          ChannelServices.RegisterChannel(_tcpServerChannel, false);
        RemotingConfiguration.RegisterWellKnownServiceType(
            typeof(CameraControllerProxy),
             "CameraControllerProxy",
             WellKnownObjectMode.Singleton);
```

```
        try
            Start () ;
        catch (Exception exc)
            Logger.LogException(exc, new LogEventArgs("CameraController"));
    }
}
```

Výpis 1: Ukázka registrace třídy vzdáleného volání

3.1.3 Implementace WCF

Knihovny vrstvy WcfServiceLibrary konzumují pomocí .NET Remoting veřejné metody hostované v naší službě *Remoted Server Service*. Úkolem těchto tříd je zprostředkovat ovládání periferií pomocí komunikačních kanálu WCF služeb. Samotné knihovny pak jsou hostovány v IIS pro přístup v rámci lokální intranetové sítě. Pokud bychom chtěli zpřístupnit služby pro použití v rámci internetu, stačí vhodně nastavit přesměrování na routeru. WCF služby pak implementují bezpečnostní opatření v podobě nutnosti autentizace příchozího požadavku oproti místnímu uživatelskému účtu systému Windows. Ověření probíhá vždy při prvním příchozím požadavku, poté IIS vygeneruje sezení, aby po dobu platnosti této session mohla komunikace probíhat rychleji. Toto chování lze změnit např. prostřednictvím IIS manageru. Každá s našich WCF služeb implementuje frontu příchozích požadavků, které postupně obslouží. Tím je zajištěna konzistence a správné chování systému.

Pro správné fungování WCF služby je třeba, aby bylo deklarováno veřejné rozhraní označené jako *[ServiceContract]*, v našem případě se jedná o rozhraní **ICameraService**, které pomocí příznaku *[OperationContract]* definují služby použitelné prostřednictvím WCF. Na základě tohoto rozhraní pak IIS generuje popisný soubor služby WSDL. Třída **CameraService** pak implementuje metodu pro získání fotografie z kamery. Nejprve zaregistrujeme klientský kanál pro .NET Remoting. V metodě pro získání fotografie se prostřednictvím tohoto kanálu zeptáme serveru, zda má objekt *CameraControllerProxy*, pokud ano, získáme jeho proxy instanci, ze které získáme pole bajtů reprezentující požadovaný snímek. Ten pře vrácením upravíme na požadovanou velikost a kvalitu.

```
[ServiceContract]
public interface ICameraService
{
    [OperationContract]
    byte[] GetActualPicture(int width, int height, ImageQuality imageQuality);
}

public class CameraService : ICameraService
{
    public CameraService()
    {
        TcpClientChannel tcpClientChannel = new TcpClientChannel();
        ChannelServices.RegisterChannel(tcpClientChannel, false);
    }
```

```
    public byte[] GetActualPicture(int width, int height, ImageQuality imageQuality)
    {
        var cameraControllerProxy = Activator.GetObject(typeof (CameraControllerProxy),
                                                CameraControllerProxyUri) as
                                    CameraControllerProxy;

        if (cameraControllerProxy != null)
        {
            byte[] imgBytes = cameraControllerProxy.ImageBytes;

            var imagesHelper = new ImagesHelper(imgBytes);
            imgBytes = imagesHelper.ResizeImageInScale(width, height, (int)imageQuality);
            imagesHelper = null;

            return imgBytes;
        }

        return null;
    }
}
//Konfigurace WCF služby pro IIS7
<service name="WCF.CameraLibrary.CameraService" behaviorConfiguration="MyBehavior">
    <host>
      <baseAddresses>
       <add baseAddress="http://localhost/WCF.CameraLibrary/CameraService/"/>
      </baseAddresses>
    </host>

    <endpoint address=""
              name="wsHttpBinding"
              binding="wsHttpBinding"
              contract="WCF.CameraLibrary.ICameraService"
              bindingConfiguration="wsHttpBindingConfig">
    </endpoint>

    <endpoint address="basic"
              name="basicHttpBinding"
              binding="basicHttpBinding"
              contract="WCF.CameraLibrary.ICameraService"/>

    <endpoint address="mex"
              binding="mexHttpBinding"
              contract="IMetadataExchange" />
</service>
```

Výpis 2: Ukázka implementace knihovny WCF služby

3.2 Implementace mobilního klienta

Mobilní klient bude ovládat periferie pomocí WCF služeb nabízených embedded serverem. V ukázce kódu se pokusím nastínit, jak z .NET CF konzumovat WCF služby. V tomto ohledu nám trochu usnadní práci Visual Studio, které je schopné z předaného

WSDL popisu vygenerovat proxy třídy. Jelikož budeme pracovat s metodami, které budou komunikovat v síti (blokující operace), rozhodl jsem se pro postup používání asynchronního volání metod. To znamená, že pokud chci získat např. obrázek, pomocí metody *GetPictureAsynch()* spustím ve vlastním vlákně skutečné volání metody. Ve chvíli, kdy vlákno získá odpověď serveru, vyvolá se událost, která oznámí všem registrovaným posluchačům, že přišla odpověď. Jako argument se předá vlastní implementace třídy dědící z *EventArgs*, která slouží jen k obalení skutečné odpovědi. To pro nás znamená, že vrstva uživatelského rozhraní se zaregistruje jako posluchač a ve chvíli kdy dorazí odpověď, zobrazí výsledek, který získá z argumentu *CameraResultEventArgs*.

```
public class CameraClient
{
    public event AsynchWcfDelegate GetPictureAsynchEvent;
    private void InvokeGetPictureAsynchEvent(CameraResultEventArgs e)
    {
        AsynchWcfDelegate handler = GetPictureAsynchEvent;
        if (handler != null) handler(this, e);
    }

    private readonly CameraServiceClient _cameraServiceClient;
    private string remoteAddress = @"http://admin-pc/WcfServiceServer/CameraService.svc";
    private string endPointName = "wsHttpBinding";

    public CameraClient()
    {
        _cameraServiceClient = new CameraServiceClient(endPointName, remoteAddress);
        _cameraServiceClient.GetActualPictureCompleted +=
            CameraServiceGetActualPictureCompleted;
    }

    private void CameraServiceGetActualPictureCompleted(object sender,
        GetActualPictureCompletedEventArgs e)
    {
        var cameraResultEventArgs = e.Error != null ?
            new CameraResultEventArgs(e.Error) : new CameraResultEventArgs(e.Result);

        InvokeGetPictureAsynchEvent(cameraResultEventArgs);
    }

    public void GetPictureAsynch()
    {
        _cameraServiceClient.GetActualPictureAsync(480, 800, ImageQuality.Full);
    }
}
```

Výpis 3: Ukázka implementace WP7 ovládání

Každá třída reprezentující klienta WCF služby je založena na návrhovém vzoru **Singelton**. To nám zaručí, že pro všechna okna UI je k dispozici pouze jedna instance, čímž zaručíme korektní chování a integritu požadavků. Pro tvorbu přívětivého uživatelského rozhraní je použita technologie Silverlight, která je postavena na jazyce XAML. Část z toho, jak vypadá klientská aplikace si můžete prohlédnout na obrázku 21.

Obrázek 21: Klientská aplikace dálkového ovládání

4 Nasazení a zhodnocení výsledku ve srovnání s konkurencí

V této kapitole popíši, jak probíhalo nasazení aplikace u klienta. Na základě zpětné vazby si zhodnotíme, jak byl klient spokojen. Poté se pokusím srovnat naší implementaci s řešeními, která jsou již dnes nabízena na českém trhu.

4.1 Nasazení našeho řešení

Abychom byli schopni posoudit, jak systém fungoval a jak intenzivně byl používán, použili jsme náš logovací systém, který zaznamenával jak chyby, tak volání jednotlivých metod. Samotné nasazení našeho řešení pro inteligentní ovládání domácnosti probíhalo v několika logických krocích.

1. Instalace hardwarových komponent. Nejprve jsme museli uložit embedded desku do ochranného obalu. Museli jsme najít vhodné místo pro umístění tak, abychom byli schopni umístit kameru tak, aby měla přímou viditelnost na venkovní dveře. Zde jsme byli limitováni maximální délkou USB kabelu kamery a délkou VGA kabelu dotykového LCD panelu. Ostatní komponenty, jako siréna, Quido ETH 4, nebo relé deska pro osvětlení místem umístění limitovány nejsou. Instalaci relé desky pro ovládáni osvětlení prováděl odborník (elektrikář).

2. Druhým krokem bylo připojení serveru a Quida ETH4 do stávající lokální sítě. Pro připojení obou zařízení s Wi-Fi routerem jsme použili přímých síťových kabelů UTP5e. Dále bylo třeba nakonfigurovat IP adresu Quida a serveru dle adresního rozsahu místní. Serveru tedy byla přiřazena IPv4 adresa 192.168.0.3 a Quidu 192.168.0.2.

3. Po zapojení všech komponent serveru jsme mohli spustit server a nakonfigurovat naši službu tak, aby se spouštěla automaticky při zapnutí serveru. Po spuštění služby začal systém fungovat.

4. Předposledním krokem byla instalace klientské aplikace dálkového ovládání na mobilní telefon klienta. Při prvním spuštění klientské aplikace došlo ke stažení konfiguračního souboru ze serveru, který popisuje umístění a typ jednotlivých periferií.

5. Nyní zbývá finální část nasazení. Zaškolení klienta pro používání řešení. Jelikož se jednalo o pilotní provoz, vysvětlili jsme klientovi také, co dělat v případě pádu systému. Ovládání aplikace je jednoduché a intuitivní, tak tato fáze nám zabrala jen několik minut.

4.1.1 Posouzení pilotního provozu

U klienta byla aplikace testována téměř dva měsíce. Služba dokázala běžet v kuse celých 28 dní, poté bylo bohužel nutné ji manuálně restartovat kvůli chybě ovladačů kamery. Služba se tedy jeví poměrně spolehlivě. Dále jsme z logu vysledovali četnost použití jednotlivých

funkcí uvedený v tabulce 7. Uvedené údaje mohou být zavádějící. Především u použití kamery musíme brát v úvahu, že se snímky získávají periodicky samy, to znamená, že počet použití rychle narůstá. Dále z měření plyne, že klient přibližně dvakrát denně pomocí ovládání rozsvěcel a zhášel světla v pokoji. Než se pustíme do dalšího hodnocení, dovolil bych si citovat slova klienta, kterého jsme požádali o krátké vyjádření.

> Ráda bych se touto cestou podělila o své praktické zkušenosti s využíváním "ovládače domu". Musím nejprve podotknout, že moje životní situace není jednoduchá – sama jsem upoutána na invalidní vozík se značným pohybovým omezením a navíc se musím postarat o svoji starou maminku, která je po mozkové mrtvici. Žijeme v sociálním byte ve dvoupatrovém rodinném dome, kde je pro nás invalidy značně ztížené prostředí. Bohužel prostředí sociálních bytu „zdravých invalidu" je narušeno jednak technickými překážkami a jednak samotným asociálním chováním ostatních nájemníku, kteří pocházejí z problémových skupin obyvatelstva. Je třeba ještě zdůraznit, že můj svět znamená vlastně pohyb na vozíku po chodbě s chabým osvětlením a dále vlastní prostor bytu. Vzhledem ke skutečnosti, že jsme dvě ženy odkázané jen na sebe, je logické, že se také obáváme o naší bezpečnost, neboť máme dennodenní zkušenosti s drogovými závislými a různými živly. Pro mne jsou důležité i takové maličkosti, které se zdravému člověkovi zdají samozřejmé nebo malicherné a to je například možnost vědět, kdo skutečně zazvonil u dveří a zda muže mít dobré či špatné úmysly. Přivítala jsem proto možnost využít „ovládač" s tím, že mi umožní nahlédnout za dveře na chodbu a prozkoumat blízké okolí a připravit mne na cestu k výtahu. Jako neocenitelnou službu považuji možnost dálkového rozsvěcení a zhášení světel v byte, neboť se to netýká jen mne, jakožto nechodícího, ale také různých životních situací, kdy v průběhu noci nemusím vynakládat tolik úsilí při vstávání k nejbližšímu vypínači, abych zhasla za svoji matkou, která trpí nespavostí a sklerózou. Ačkoliv patřím mezi lidi již středního veku dnešní doba mne naučila přizpůsobit se, a tak vlastní obsluha mobilu s tímto dálkovým ovládáním mi nečinila po krátkém zaškolení vetší problém. Určitě bych uvítala kdyby se dálkově daly otvírat i balkónové dveře. Určitou nevýhodou je cena. Samozřejmě jsem se ptala, zda bude toto zařízení prodejné, ale cena blížící se 60 000 Kč je pro mne bohužel příliš vysoká. Otázkou zastává, zda by tento produkt nemohl být dotován, naprosto stejně, jako byl i můj elektrický vozík.
> Marie Grundelová

Klient si také stěžoval na delší reakční dobu při použití mobilního ovládání, to nás přimělo ke změně hardwaru a operačního systému, jak jsem již popisoval dříve. Ačkoli je hodnocení klienta převážně pozitivní, dává nám podmět k dalšímu zlepšování systému. Co se týče výtky absence dálkového ovládání dveří, naše řešení samozřejmě umožňuje připojení elektronických zámků, ale jelikož se jednalo o první větší test, neměli jsme odvahu tento prvek nasadit. Pro ovládání zámku by bylo využito stejné desky, jako pro ovládání světel, pouze s tím rozdílem, že by místo rozsvícení světla došlo k uvolnění zámku.

periferie	celková četnost	průměrně za den
Světlo v pokoji	271	4,5
Světlo v chodbě	117	1,95
Siréna	17	0,28
Kamera	310	5,18

Tabulka 7: Četnost využití služeb systému při pilotním provozu

4.2 Existující konkurenční řešení

Na českém trhu se nachází několik společností zabývajících se návrhem a implementací automatizovaných nebo inteligentních domů. V této kapitole porovnám několik z nich a pokusím se klady a zápory těchto řešení. Ani jedno z řešení, které jsem našel, se neorientovalo na doménu řešení automatizace pro osoby s postižením, ať už fyzickým nebo jiným. Při průzkumu trhu jsem však narazil na neochotu sdělovat, nebo dokonce veřejně vystavovat technické parametry, použité technologie, nebo cenu svých řešení. Většina firem tedy jedná s klientem jednotlivě a až na základě požadavků tvoří finanční analýzu. Nicméně jsme zjistili, že se nabízejí dvě varianty řešení.

4.2.1 Insight home

Prvním zástupcem je tzv. řešení Insight home. Jedná se o řešení nasazované v moderních domech (bytech) a nabízí komplexní řešení pro pohodlný životní styl. Zahrnuje ovládání domácích multimediálních zařízení, jako Tv, BlueRay přehrávače, Hi-Fi systémy. Dále nabízí integraci bezpečnostních prvků, jako kamery a alarmy. Pro vetší pohodlí umožňuje ovládání osvětlení, topení a dalších periferií. Tato společnost má k dispozici dva vlajkové domy, kam se můžete jít podívat, jak jejich technologie fungují. Bohužel se areál nachází až v Praze a nepodařilo se nám domluvit si schůzku na vhodný termín.

Podle uvedených referencí jsme zjistili, že jejich realizace umožňující řízení světel, žaluzií, vytápění, kamerového systému, měření spotřeb energií, audia a videa pomocí dotykového panelu nás vyjde na nemalou částku blížící se k 600 000 Kč. To je z uvedených projektů nejnižší částka.

Společnost uvádí, že je schopna realizovat řešení během několika dnů. Tato doba samozřejmě závisí na tom, zda se rozhodnete pro jejich řešení v době, kdy si zrovna stavíte dům a tak je umožněn přístup k rozvodům, nebo až v době, kdy je dům již postaven a řešení chceme teprve dodatečně integrovat.

4.2.2 Haidy

Druhé řešení, které je dostupné na našem trhu nabízí společnost Haidy. Haidy staví své řešení na několika pilířích.

- Komfort - HAIDY neustále monitoruje měřitelné veličiny ve Vašem domě v jeho okolí a na základě těchto vstupů v něm spouští optimální akce. Například rozsví-

cení světel můžete navázat na detektory pohybu, na senzory úrovně světelnosti, případně na určitý čas.

- Bezpečnost - Integruje v sobě různé typy detektorů (pohybové, požární, rozbití skla, úniku plynu aj.), nebo kamerový systém. HAIDY umí také obsloužit elektronické uzamykání domu.
- Spolehlivost, dostupnost a flexibilita - V případě výpadku některé části HAIDY nemůže dojít ke kolapsu celého systému. Díky své konfigurovatelnosti neboli širokým možnostem nastavení, může být systém HAIDY „ušit na míru“ každému zákazníkovi a přizpůsoben každému rozpočtu.

Následující obrázek převzatý z veřejně dostupného katalogu uvádí střední variantu jejich instalace. Cena řešení je pak vyčíslena na 499 827,-

Kategorie	druh	umístění	počet
Osvětlení	**spínaná světla**		**25**
	stmívaná světla		6
Ovládání	**Jednoduchý vypínač**		**2**
	Dvojitý vypínač		34
Topení	**ovládání servoventilů**	**tepelné rozvaděče**	**12**
Elektronický Zabezpečovací Systém	**Detektor pohybu**		**15**
	Magnetický kontakt		29
Měřiče	**Vodoměr**		**1**
	Elektroměr		1
Komunikace	**GSM brána**		**1**
	Videovratný		1
Ostatní	**elektrozámek**	**branka**	**1**

Obrázek 22: Cena HAIDY řešení

4.2.3 Srovnání jednotlivých řešení

Obě tyto představené varianty nabízí velice komplexní řešení pro nejrůznější možnosti ovládání prvků v domácnosti. Za jejich řešeními stojí mnohaletá zkušenost v oblasti tvorby automatizovaných řešení. Podívejme se ale na finanční náklady pro pořízení řešení nabízející téměř stejnou funkcionalitu, jako naše dosavadní řešení (tj. ovládání světel, kamery, teplotních čidel, elektronického zámku, přítomnost dálkového ovládání). Naše řešení (obrázek 23) je možné pořídit za téměř desetinu ceny, kterou uvádí konkurence.

Zařízení	Počet kusů v řešení	Jednotková cena	Souhrnná cena
Návrh hardware			
JetWay NF 96	1	2300 Kč	2300 Kč
Dotykový panel	1	12000 Kč	12000 Kč
PC komponenty	1	2000 Kč	2000 Kč
Quido ETH 8/8	1	5 280 Kč	5280 Kč
Wifi Router	1	560 Kč	560 Kč
Ip kamera	1	4749 Kč	4749 Kč
Elektrický zámek	1	1560 Kč	2560 Kč
Boxy	2	900 Kč	1800 Kč
Instalace			
Prvotní zapojení			10 000 Kč
Test zařízení			3000 Kč
Náklady na mobilní telefon			
HTC Mozart			6638 Kč
Náklady celkem			
			55 886 Kč

Obrázek 23: Kalkulace nákladů našeho řešení

5 Závěr

Tento projekt byl velice zajímavý tím, že se jedná o reálný projekt, který má navíc honosné poslání pokusit se usnadnit život tělesně postiženým lidem. Pro uchopení projektu a zjištění potřeb jsme se myslím velice vhodně opřeli o metodiku UP, která nám poskytla „návod", jak zjisti skutečné potřeby klienta, se kterým jsme byli v kontaktu.

Nejprve by tím, co si myslím, že se této práci prozatím povedlo. Povedlo se nám vytvořit robustní, ale přehlednou modulární architekturu, která skýtá značný potenciál pro budoucí rozšíření o další ovladatelné periferie. Díky kvalitnímu vícevrstvému návrhu můžeme velmi snadno optimalizovat, nebo dokonce vytvářet nové verze uživatelského rozhraní s ohledem na druh postižení, a tak i různorodé potřeby. Dále bych si zde dovolil vyzdvihnout API pro komunikaci komunikaci s vrstvou představující hardware, které je velmi dobře a jednoduše použitelné jak z aplikací běžících přímo na serveru, jako náš plánovač úloh, nebo například z aplikací sloužících jako dálkové ovládáni přes Wi-Fi. V případě potřeby je možné vystavit toto rozhraní i pro využití přes internet. Tak si můžete například na dovolené monitorovat stav domácnosti.

Jak jsem již zmínil, aplikace skýtá veliký potenciál pro nasazení dalších ovládaných prvků domácnosti pro které je připravena. Bohužel díky finanční náročnosti na pořízení těchto periferií jsme si nemohli dovolit je vyzkoušet v takovém množství a rozsahu, jak bychom si přáli. Zde vidím do budoucna prostor pro zlepšení. Na druhou stranu jsme však udrželi celkové náklady ne realizaci na velmi rozumné hladině.

Rád bych zde ještě jednou zmínil skrze pozitivní hodnocení klienta, který byl po dobu testování s aplikací spokojen. Jistě si vezmeme k srdci i jeho výtky, které převážně spočívaly v absenci některých nákladnějších ovládacích prvků.

Celkově si tedy myslím, že se práce povedla a pevně věřím, že v krátké době a s drobnými úpravami, se nám podaří toto řešení začít prodávat. Již teď jsme v kontaktu s hotelem, kterému jsme v minulosti vytvářeli informační systém. Majitel hotelu zvažuje nasazení našeho řešení za účelem ovládání osvětlení a vytápění pokojů z recepce.

6 Reference

[1] Arlow, Jim, *UML2 a unifikovaný proces vývoje*, Brno: Computer Press, 2008, ISBN 978-80-251-1503-9

[2] Robinson, Simon, *C# Programujeme profesionálně*, Brno: Computer Press, 2003, ISBN 80-251-0085-5

[3] Peiris, Chris and Mulder, Dennis, *Pro WCF: Practical Microsoft SOA Implementation*, New York: APress, 2007, ISBN 978-1-59059-702-6

[4] YAO, Paul, *Programming .NET Compact Framework 3.5 (2nd Edition)*,Addison-Wesley Professional, 2009, ISBN 978-0321573582

[5] Insight Home, *http://www.insighthome.eu/reference.html*, [Online] 3.2.2011

[6] HAIDY, *http://www.haidy.cz/stredni-dum-optimal*, [Online] 2.2.2011

[7] Papouch, dokumentace Quido ETH, *http://www.papouch.com/cz/shop/product/quido-rs-4-4-vstupy-vystupy-a-teplomer-na-232-485/*, [Online] 20.10.2010

[8] Plucar, Jan, *Softwarové rozhraní inteligentního domu*, diplomová práce, VŠB-TU FEI, Ostrava, 2011

A Přílohy k diplomové práci

A.1 Specifikace Ubee dongle

CD\Přílohy\umubee.pdf

A.2 Specifikace protokolu Spinel

CD\Přílohy\quido-spinel.pdf

A.3 Diagramy tříd aplikací

CD\Přílohy\Třídní diagramy\